"The Surface Model"
An Uncertain Continuous Representation of the Generic Camera Model and its Calibration

Von der

Carl-Friedrich-Gauß-Fakultät

der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines

Doktoringenieurs (Dr.-Ing.)

genehmigte Dissertation von

Dipl.-Ing. Dennis Rosebrock

geboren am 26.12.1980

in Peine

Eingereicht am: 06.08.2015

Disputation am: 19.02.2016

1. Referent: Prof. Dr.-Ing. Friedrich M. Wahl

2. Referent: Prof. Dr.-Ing. Dr. h.c. mult. Wolfgang Förstner

2016

Acknowledgements

With joy I think back at my time in the Institute of Robotics of the Technical University of Braunschweig. Apart from writing my thesis, I worked on many other things, which sometimes were even more fun. I got to develop autonomous cars (and see them actually driving by themselves), visit foreign countries to present my work, and hold lectures in front of a crowd of 200 more or less interested students. But even more importantly, I made new friends at the institute. I will never forget the discussions in our little "coffee kitchen" where we, while drinking tea, developed with enthusiasm new groundbreaking ideas like self-deactivating squid robots. And of course the activities we had outside of our building (having to deal with the sun and all), enriched my personal life.
I would like to thank my advisor and Head of the Institute of Robotics, Professor Friedrich M. Wahl. He gave me the possibility to explore any topic of my personal interest. This allowed me to gain knowledge and experience in many different areas, such as computer vision, robotics, advanced driver assistance systems and, last but not least, camera calibration. I am deeply thankful for all the freedom Professor Wahl gave me, his helpful advice and for his trust and support, which continued even after his retirement.
Many thanks go also to my second advisor, Professor Wolfgang Förstner. His publications and the fruitful discussions we had allowed me to gain new insights into the topic of parameter estimation and therefore lead to the development of the approach proposed in this thesis.
Additionally, I was very grateful that Anne-Sophie Poulin-Girard and her team provided calibration data, such as I was able to verify my approach for their extraordinary panomorph camera lens.
Without the help of my proof-readers, who gave a lot of useful hints and advice, this work would be significantly less structured and more cumbersome to read. Thank you very much Grace, Dirk, Caro and Volker for your help and your time.
Finally, I thank those who believed in me and my abilities to make a scientific contribution. This includes my family, my colleagues, my friends and everyone who supported me and helped me to never lose sight of the goal of finishing this work.

Dennis Rosebrock

Abstract

Using digital cameras for measurement purposes requires the knowledge of the mapping between 3D world points and 2D positions on the image plane. There are many different mathematical models that provide this mapping for a specific imaging system. These models tend to make assumptions about the structure of the system, e.g. an exact alignment of vision sensor and lenses or mirrors. If these constraints are not met or an inappropriate model is chosen, the measurement process will eventually deliver inaccurate results.
To avoid these problems, Grossberg and Nayar proposed a discrete generic camera model that describes a digital camera by assigning an arbitrary viewing ray to each pixel of the camera image. This makes the model applicable to any kind of camera, especially also to non-central ones like onmidirectional catadioptrics. However, this model is difficult to use in practice, as there is no direct method for mapping a 3D point to the image or determining rays for subpixel image positions.
In this work, the *Surface Model*, an uncertain continuous representation of the generic camera model, will be introduced. It uses a spline surface in 6D Plücker space to describe the camera. The interpolation abilities of the spline surface allow the viewing ray and its uncertainty for any (subpixel) position to be easily determined. Furthermore, the description facilitates the mapping from 3D world points to the image.
The calibration of the generic model has to be performed pixel-wise and is technically involved and time-consuming. In this work, hand-held sparse planar chessboard patterns are used for calibration. This introduces the assumption of a certain mapping continuity, but the calibration is much simpler to execute from a technical point of view. Furthermore, the uncertainties of the corresponding image point measurements are taken into account and propagated during the complete calibration procedure to obtain an uncertain model. It delivers uncertainty information in the form of covariance matrices for each camera operation. Simulations prove the validity of each step and the practical applicability of the procedure is shown by calibrating several real cameras of different types.

Kurzfassung

Um digitale Kameras zu Vermessungszwecken einzusetzen muss der mathematische Zusammenhang zwischen 3D Weltpunkten und 2D Bildpunkten bekannt sein. Es existiert eine Vielzahl an mathematischen Modellen, welche diese Abbildung für spezifische Kamerasysteme beschreiben. Für deren Gültigkeit ist die Einhaltung der zugehörigen Randbedingungen, beispielsweise die hochgenaue Ausrichtung von Bildsensor, Linsen und Spiegeln, zwingend erforderlich. Andernfalls können stark fehlerhafte Messergebnisse die Folge sein. Um diese Problematik zu meiden, haben Grossberg und Nayar ein diskretes generisches Kameramodell vorgeschlagen. Dieses zeichnet sich dadurch aus, dass jedem einzelnen Pixel ein separater Sehstrahls zugeordnet wird. Somit kann jede erdenkliche Kamera beschrieben werden. Dies gilt auch, wenn kein punktförmiges optisches Zentrum existiert, wie es zum Beispiel bei omnidirektionalen catadioptrischen Systemen der Fall sein kann. Aufgrund der Diskretisierung kann allerdings nicht für jede beliebige Subpixel-Position ein Sehstrahl ermittelt werden kann. Auch die Projektion eines beliebigen 3D-Punktes ins Kamerabild ist nicht ohne Weiteres möglich.

In dieser Arbeit wird das *Surface Model* vorgestellt. Es dient als eine kontinuierliche Repräsentation des generischen Kameramodells, welche Modellunsicherheiten explizit berücksichtigt. Zur mathematischen Beschreibung wird eine Splineoberfläche im sechsdimensionalen Plücker-Raum genutzt. Deren Interpolationsfähigkeiten erlauben es, für jedwede Subpixel-Position direkt einen Sehstrahl zu ermitteln, sowie einen beliebigen 3D-Punkt unmittelbar ins Kamerabild zu projizieren.

Die Kalibrierung des diskreten generischen Modells erfordert die Bestimmung mehrerer Messpunkte für jeden einzelnen Pixel. Entsprechende Verfahren sind zeitaufwändig und technisch anspruchsvoll. Um den Prozess zu vereinfachen, werden in dieser Arbeit von Hand platzierte, planare Schachbrettmuster eingesetzt.

Während der Messdatengewinnung für die Kalibrierung treten unweigerlich Messungenauigkeiten auf. Beim hier vorgestellten Verfahren zur Parameterermittlung des Surface Models werden diese Unsicherheiten explizit zur Stabilisierung und Verbesserung der Genauigkeit genutzt. Dies resultiert in einem unsicheren Kameramodell, welches für die Anwendungen der Sehstrahlermittlung und der Punktprojektion Ergebnisunsicherheiten in Form von Kovarianzmatrizen zur Verfügung stellt.

Mittels Simulationen wird die Anwendbarkeit sämtlicher vorgestellter Verfahren validiert. Durch die Kalibrierung verschiedener realer Kameras wird darber hinaus deren praktische Nutzbarkeit aufgezeigt.

Contents

Notation

General	
p	scalar
$\boldsymbol{x}$	general inhomogeneous vector
$\mathbf{x}$	general homogeneous vector
$\mathbf{x}^s$	spherically normalized homogeneous vector (unity length)
$\boldsymbol{p} = [p_1\ p_2]^T$	inhomogeneous 2D point
$\mathbf{p} = [p_1\ p_2\ 1]^T$	homogeneous 2D point
$\boldsymbol{P} = [p_1\ p_2\ p_3]^T$	inhomogeneous 3D point
$\mathbf{P} = [p_1\ p_2\ p_3\ p_4]^T$	homogeneous 3D point
$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$	general matrix
I_n	$(n \times n)$ identity matrix
0_n	$(n \times n)$ matrix of zeros
$\mathbf{0}_n$	vector of zeros with n elements
$\mathsf{S}(\boldsymbol{P})$	(3×3) skew symmetric matrix from elements of 3D point $\boldsymbol{P}$, Section 2.1.2
$\mathbf{L} = [\boldsymbol{d}^T\ \boldsymbol{m}^T]^T$	Plücker line coordinates with direction vector $\boldsymbol{d}$ and moment vector $\boldsymbol{m}$, Section 2.1.3
$\overline{\Gamma}$	dual Plücker matrix (2.1.8)
R	(3×3) rotation matrix
$\boldsymbol{t}$	3D translation vector
${}^1\mathrm{T}_2$	(4×4) homogeneous transformation from reference coordinate system 1 to coordinate system 2
$\boldsymbol{R}$	Rodrigues vector (Section 2.1.4.2)
$\mathrm{J} = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}$	Jacobi matrix containing the partial derivatives of a multidimensional function $\boldsymbol{f}(\boldsymbol{x})$ with respect to the elements of the parameter vector $\boldsymbol{x}$

Probabilistics	
l_i	single measurement
$\tilde{l}$	true value
$\widehat{l}$	estimated value
$\boldsymbol{l}$	vector of measurements
$\Sigma_{\boldsymbol{xx}}$	covariance matrix of an inhomogeneous vector $\boldsymbol{x}$
$\Sigma_{\mathbf{xx}}$	covariance matrix of a homogeneous vector $\mathbf{x}$
$\mathsf{N}(\bar{\boldsymbol{x}}, \Sigma_{\boldsymbol{xx}})$	multivariate normal distribution with mean value $\bar{\boldsymbol{x}}$ and covariance matrix $\Sigma_{\boldsymbol{xx}}$
B-splines	
u, v	spline parameters
p	spline degree
u_i, v_i	spline knots
s_i	span i of a spline curve
$\boldsymbol{C}(u)$	spline curve function
$\boldsymbol{S}(u, v)$	spline surface function
$\boldsymbol{P}_i$	spline curve control points (arbitrary dimension)
$\boldsymbol{P}_{ij}$	spline surface control points (arbitrary dimension)
$n+1$	number of control points in u direction
$m+1$	number of control points in v direction
$\boldsymbol{Q}_k$	data points for curve creation
$\boldsymbol{Q}_{kl}$	data points for surface creation
Estimation	
$\boldsymbol{f}(\boldsymbol{x})$	model function
$\boldsymbol{x}$	parameter vector
$\tilde{\boldsymbol{x}}$	true parameters
$\widehat{\boldsymbol{x}}$	estimated parameters
$\boldsymbol{l}$	measurement vector
$\tilde{\boldsymbol{l}}$	true measurements
$\widehat{\boldsymbol{l}}$	estimated measurements
$\tilde{\boldsymbol{v}}$	true measurement corrections
$\widehat{\boldsymbol{v}}$	estimated measurement corrections
$\widehat{\boldsymbol{x}}^a$	approximation of the estimated parameters
$\widehat{\boldsymbol{l}}^a$	approximation of the estimated measurements
$\widehat{\boldsymbol{v}}^a$	approximation of the estimated corrections
A	design matrix or matrix of partial derivatives for parameters
B	matrix of partial derivatives for measurements
W	weight matrix
Ω	Mahalanobis distance
$\Sigma_{\boldsymbol{ll}}$	covariance matrix of the measurements
$\Sigma_{\widehat{\boldsymbol{x}}\widehat{\boldsymbol{x}}}$	covariance matrix of the estimated parameters

Chapter 1

Introduction

Digital cameras are widely used for general measurement tasks in the field of computer vision. Common applications are taking 3D measurements of the environment via multi-camera systems and determining the movement of a single camera by using the concepts of *visual odometry*. The combination of these methods obtains camera movement and the 3D structure of the environment from consecutive images of a single camera at the same time and is called *structure from motion* (SfM). This procedure is widely used in different fields that utilize cameras as a primary sensor, including photogrammetry, mobile robotics and also advanced driver assistance systems (ADAS) in the area of automotive applications.
To implement SfM for an imaging systems, it is crucial to know its exact internal structure. This structure needs to be represented by a mathematical model that describes the relation between the three-dimensional world and the two-dimensional camera image. Generally, two main functions are needed: *forward projection* and *back projection* (see Figure 1.0.1). Forward projection calculates the 2D image position that a 3D world point will be mapped to. Back projection, on the other hand, is the calculation of the *viewing ray* for a specified image position. Said differently, it delivers a mathematical description (usually a single line) of all the points in 3D space that can be seen at that pixel. With

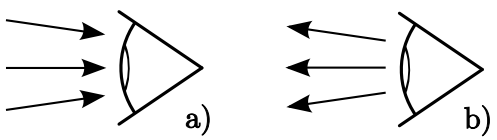


Figure 1.0.1.: a) Forward projection projects elements in the 3D world to image coordinates. The projection direction is *towards the camera*. b) Back projection gives the 3D viewing ray for a specific image position. The projection direction is *away from the camera*.

the help of these two functions, a wide range of measurement tasks can be taken with camera-based imaging systems. Furthermore, optimization algorithms can be executed to refine the measurements and compensate for errors.
As mentioned before, knowing the internal composition of an imaging system is essential to achieve highly accurate measurement results. If the physical structure would be perfectly known, the mathematical model could be created by utilizing the laws of optics. However, these systems tend to be composed of multiple different parts, including various optical lenses and mirrors, which makes the derivation of an exact physical model a challenging problem. Therefore, much simpler formulations are commonly used to describe the camera mappings, i.e. forward and back projection, with a sufficient accuracy. These models are usually designed for a specified type of imaging system and make certain assumptions. Commonly it is assumed that the camera is *central*, i.e. that an optical center exists where all viewing rays intersect. Although this assumption is valid for simple dioptric systems, which use lenses to focus the incoming light in a single point, it is generally not the case with more complex omnidirectional setups that may incorporate mirrors of different shapes. To describe such *non-central* systems, a much more complex model is necessary. The first step when using a digital camera for measurement purposes is therefore to choose an adequate model. Choosing the best model requires expert knowledge of both the inner structure of the camera, as well as the available models that can be used to accurately describe it. One contribution of this work is to provide a model that can be used to represent many different kinds of camera systems, including central and non-central ones, and provide the basic functionalities of general forward and back projection. This spares the user of a digital camera from the tedious task of model selection. The model proposed in this thesis is a continuous representation of the generic camera model introduced by Grossberg and Nayar in [GN01]. This new representation uses spline surfaces as a mathematical description and will therefore be called "*The surface model*".
After choosing an appropriate camera model, its parameters have to be determined. This process is called *camera calibration* and is usually specifically designed for the selected model. The calibration process is very complex and relies on the utilized hardware materials meeting certain requirements. In general, calibration objects with exactly known sizes must be used. These objects have to be three-dimensional, which makes it more difficult to construct them. If the objects are moved during the calibration procedure the relative translation has to be exactly known. Sometimes, even more complex hardware like digital displays are needed. This can make the procedure very cumbersome or even impossible to execute for users who lack the necessary equipment. An easier method is to use planar sparse calibration patterns, which can easily be produced, and to place them by hand in different positions in front of the camera. This was first proposed by Zhang in [Zha02] for the calibration of central cameras with minor distortions and later adapted by Sturm and Ramalingam in [SR04] to determine the parameters of the generic camera model of Grossberg and Nayar. The second contribution of this work is the proposal of a calibration procedure for the surface model presented in this work, which utilizes sparse planar calibration objects. Since these objects can be placed by hand, the calibration

process is greatly simplified for the user. By utilizing spline surfaces for interpolation purposes, highly accurate results can be achieved. Furthermore, the calibration process is independent of the type of the camera system, i.e. different than Sturm's method, the same procedure can be applied to central as well as non-central setups.
A commonly neglected aspect of camera calibration is the issue of measurement errors. This leads to mathematical models that are assumed to be free of any uncertainties. In reality, however, all input data are subject to noise. Therefore, as the third contribution of this work, a camera model with uncertainty information will be provided. This helps its users to assess the quality of the obtained measurements or to generate weights for a subsequent optimization procedure. Measurement uncertainties will be propagated through the complete calibration procedure and finally lead to an uncertain surface model. Therefore, the results of general forward and back projection are both accompanied by covariance matrices that describe the expected uncertainty.

1.1. Contributions

To sum up the main contributions:

1. The surface model as a continuous representation of the generic camera model is introduced, allowing to execute the tasks of general forward and back projection.

2. A calibration procedure based on hand-held sparse planar calibration patterns is proposed.

3. Measurement noise is explicitly taken into account, which provides the user of the final model with uncertainty information.

1.2. Outline

The outline of this work is as follows:

- In Chapter 2, the theoretical basics necessary to achieve the goals of this work are introduced. These include remarks on notation as well as specific mathematical concepts, such as Plücker line representations, rotations, uncertainty propagation and B-splines. Furthermore, least squares optimization and maximum likelihood estimation with homogeneous coordinates are presented.

- Chapter 3 gives an overview on camera modeling and calibration. The most common models are introduced, starting with the pinhole camera. Next, insights on omnidirectional setups are presented, followed by methods for generic camera modeling. Some of the concepts will also be used for creating and calibrating the surface model proposed in this work.

- The surface model itself is introduced in Chapter 4. It is shown how a spline surface in 6D Plücker space can be used as a continuous representation of the generic camera model. Furthermore, the basic functions of general forward and back projection are derived for this model. Simulations are used to assess the accuracy of the proposed methods. It is also shown how measurement uncertainties can be incorporated into the model, providing an uncertain representation in the end.

- Calibration of the surface model can be achieved by the method proposed in Chapter 5. Three different steps are described that lead to the parameters of the final model. An initial calibration method, which includes a bundle adjustment procedure for refinement, allows a small part of the camera image to be calibrated (Section 5.1). The results are then used as a starting point to achieve a complete calibration of the imaging system (Section 5.2). In Section 5.3 it is shown, how the continuous surface model is created. All three steps make use of uncertainty information and are validated by simulations.

- The concepts developed in the previous two chapters are used to calibrate three different real imaging systems in Chapter 6. As a result, surface models for a camera with a fisheye lens, a non-central setup composed of a dioptric camera and an omnidirectional mirror and a so-called *panomorph camera* are created and presented in this chapter.

- Chapter 7 concludes this work and gives a short outlook on further improvements of the model and possible practical applications.

Chapter 2

Theoretical basics

In this chapter, notations and the basic mathematical concepts used throughout this work will be introduced. This includes geometric entities for representing points, lines and rotations as well as the principles of uncertainty propagation, B-splines and maximum likelihood estimation in reduced homogeneous spaces. The intention is to give the reader a comprehensive list of the tools which are applied to achieve the goal of generic camera modeling and calibration. To keep the chapters with the main contributions as compact as possible, it will often be referred to the methods and examples given here. Therefore, it has mainly reference purposes and is not intended as a concise description of any of the mentioned topics. An introduction to the field of camera modeling and calibration can be found in Chapter 3, the contributions of this work are described in subsequent chapters.

2.1. Geometry

The current section will elaborate on some geometric entities that are used in this work and also introduce their notation. The main focus lies on homogeneous representations, as they play an important role in the procedures developed later on.

2.1.1. Points

In this work, bold serif letters are used for general vectors, e.g. $\boldsymbol{f}$, $\boldsymbol{y}$, $\boldsymbol{C}$. Points in 2D space are always shown by small letters $\boldsymbol{p} = (x_p, y_p)^T$, whereas capital letters are commonly used

in 3D space: $\boldsymbol{P} = (x_p, y_p, z_p)^T$. Upright letters signify a homogeneous representation, e.g. $\mathbf{p} = (x_p, y_p, 1)^T$ or $\mathbf{P} = (\lambda x_p, \lambda y_p, \lambda z_p, \lambda)^T$.

2.1.2. Skew symmetric matrix of a 3D vector

The skew symmetric matrix of a three-dimensional vector $\boldsymbol{x}$ is defined as:

$$\mathrm{S}(\boldsymbol{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{2.1.1}$$

It serves to replace the vector cross product by a matrix multiplication:

$$\boldsymbol{x}_1 \times \boldsymbol{x}_2 = \mathrm{S}(\boldsymbol{x}_1)\boldsymbol{x}_2.$$

2.1.3. Plücker line coordinates

There are various ways to represent lines in three dimensional space. The one that will be used in this work is the Plücker representation which consists of a homogeneous 6D vector

$$\mathbf{L} = [L_1\ L_2\ L_3\ L_4\ L_5\ L_6]^T. \tag{2.1.2}$$

$\mathbf{L}$ is a homogeneous representation, therefore all coordinates $\lambda\mathbf{L}$ with $\lambda \neq 0$ stand for the same line. To derive the line coordinates from two inhomogeneous 3D points $\boldsymbol{X}$ and $\boldsymbol{Y}$, it is convenient to split the contents of $\mathbf{L}$ into two parts:

$$\mathbf{L} = [\boldsymbol{d}^T\ \boldsymbol{m}^T]^T. \tag{2.1.3}$$

Here $\boldsymbol{d}$ is the line direction and $\boldsymbol{m}$ defines the so-called moment of the line (see Figure 2.1.1 for an illustration). They are calculated as

$$\boldsymbol{d} = \boldsymbol{Y} - \boldsymbol{X}, \quad \boldsymbol{m} = \boldsymbol{X} \times \boldsymbol{Y}. \tag{2.1.4}$$

From (2.1.4) it can be concluded that $\boldsymbol{d}$ and $\boldsymbol{m}$ are orthogonal. This leads to the so-called *Plücker constraint*

$$\boldsymbol{d}^T\boldsymbol{m} = L_1L_4 + L_2L_5 + L_3L_6 = 0. \tag{2.1.5}$$

Only vectors that fulfill this constraint are valid representations of a line in 3D space. Therefore, there exists a 5D subspace in 6D space that contains all valid line vectors.
If a general 6D vector $\mathbf{L}' = (\boldsymbol{d}'^T, \boldsymbol{m}'^T)^T$ is to be interpreted as a Plücker vector, the constraint (2.1.5) has to be enforced. This can be done by orthogonalization of $\mathbf{L}'$ as proposed by Förstner:

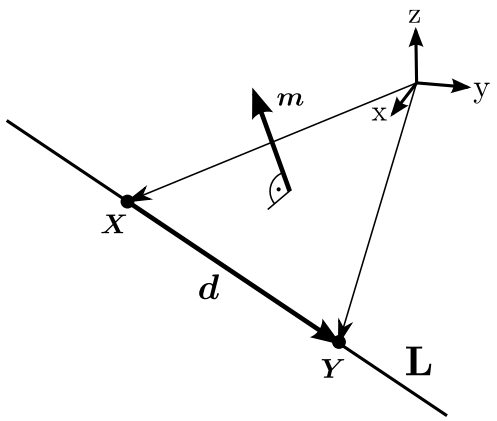


Figure 2.1.1.: The Plücker line vector $\mathbf{L}$ consists of the direction $\boldsymbol{d}$ and moment $\boldsymbol{m}$, which can be determined from two points $\boldsymbol{X}$ and $\boldsymbol{Y}$ on the line as specified in (2.1.4).

1. Get normalized direction $\boldsymbol{d}_n = \frac{\boldsymbol{d}'}{|\boldsymbol{d}'|}$ and normalized moment $\boldsymbol{m}_n = \frac{\boldsymbol{m}'}{|\boldsymbol{m}'|}$.

2. Determine distances $d = |\boldsymbol{d}_n - \boldsymbol{m}_n|$ and $r = \sqrt{1 - d^2/4}$

3. Use these elements to calculate the desired orthogonal direction and moment

$$\boldsymbol{d} = ((d + 2r)\boldsymbol{d}_n + (d - 2r)\boldsymbol{m}_n)\,\frac{|\boldsymbol{d}'|}{2d} \tag{2.1.6}$$

$$\boldsymbol{m} = ((d - 2r)\boldsymbol{d}_n + (d + 2r)\boldsymbol{m}_n)\,\frac{|\boldsymbol{m}'|}{2d} \tag{2.1.7}$$

   which form the final line coordinates $\mathbf{L} = (\boldsymbol{d}^T, \boldsymbol{m}^T)^T$.

The homogeneous character of Plücker coordinates allows to execute a normalization procedure without changing the actual line. Two of them will be used in this work:

1. *Spherical normalization* sets the length of $\mathbf{L}$ to unity, i.e. $\mathbf{L}^s = \frac{\mathbf{L}}{|\mathbf{L}|}$.

2. *Euclidean normalization* sets the length of the direction part $\boldsymbol{d}$ to one, i.e. $\mathbf{L}^e = \frac{\mathbf{L}}{|\boldsymbol{d}|}$.

When using Plücker line coordinates, certain geometric relations can be defined easily without introducing further parameters. This simplifies the construction of equation systems. Some examples and properties are given in the following list.

- Line-point incidence: the elements of a Plücker vector can be used to construct the

dual Plücker matrix

$$\overline{\Gamma} = \begin{bmatrix} 0 & L_3 & -L_2 & -L_4 \\ -L_3 & 0 & L_1 & -L_5 \\ L_2 & -L_1 & 0 & -L_6 \\ L_4 & L_5 & L_6 & 0 \end{bmatrix} \tag{2.1.8}$$

which has the convenient property that its product with a homogeneous point $\mathbf{P}_l$ that lies on the corresponding line is the zero vector:

$$\overline{\Gamma}\mathbf{P}_l = \mathbf{0}. \tag{2.1.9}$$

- Line-plane incidence: intersecting a line $\mathbf{L}$ with a plane defined by its homogeneous coordinates $\mathbf{V} = (v_1, v_2, v_3, v_h)^T = (\boldsymbol{v}^T, v_h)^T$ delivers the point of incidence

$$\mathbf{P} = \begin{pmatrix} \boldsymbol{v} \times \boldsymbol{m} - \boldsymbol{d} v_h \\ \boldsymbol{v}^T \boldsymbol{d} \end{pmatrix}. \tag{2.1.10}$$

- Line translation: shifting a line by a vector $\boldsymbol{t}$ is done by multiplication with the transformation matrix

$$\mathrm{H}_T = \begin{bmatrix} \mathrm{I}_3 & 0_3 \\ \mathrm{S}(\boldsymbol{t}) & \mathrm{I}_3 \end{bmatrix} \tag{2.1.11}$$

where I_3 is the (3×3) identity matrix, 0_3 a (3×3) matrix of zeros and $\mathrm{S}(\boldsymbol{t})$ the skew matrix of $\boldsymbol{t}$ (see 2.1.2). This gives the translated version of a line $\boldsymbol{L}$

$$\mathbf{L}' = \mathrm{H}_T \mathbf{L}. \tag{2.1.12}$$

- When direction and moment vector are exchanged, the *dual Plücker coordinates* $\overline{\mathbf{L}}$ of a Plücker vector are obtained:

$$\overline{\mathbf{L}} = [\boldsymbol{m}^T \ \boldsymbol{d}^T]^T. \tag{2.1.13}$$

2.1.4. Rotations

Rotations in 3D space can be represented by at least three parameters. When it comes to usability, however, it is sometimes beneficial to use redundant representations. Throughout this work, two different representations of rotations will be used, namely the rotation matrix R and the Rodrigues vector $\boldsymbol{R}$.

2.1.4.1. Rotation matrix R

A matrix R that belongs to the special orthogonal group $SO(3)$ is parameterized as

$$\mathrm{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = [\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3]. \tag{2.1.14}$$

It has orthogonal column vectors $\boldsymbol{r}_i$ of length 1 and the convenient property that its transpose is also its inverse:

$$\mathrm{R}^{-1} = \mathrm{R}^T = \begin{pmatrix} \boldsymbol{r}_1^T \\ \boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{pmatrix}. \tag{2.1.15}$$

The columns of R can also be interpreted as the basis of a coordinate system, given in a Cartesian reference coordinate system. Multiplying a point $\boldsymbol{P}_2$ (given in coordinate system 2) with R (defined in coordinate system 1) gives its coordinates in coordinate system 1:

$$\boldsymbol{P}_1 = \mathrm{R}\boldsymbol{P}_2. \tag{2.1.16}$$

As R has 9 parameters, it is a redundant representation. When determining these parameters from an equation system, care has to be taken that the result is indeed a member of $SO(3)$.
Rotation matrices are also commonly used in combination with a translation vector $\boldsymbol{t}$ to form a homogeneous transformation matrix. This gives

$${}^1\mathrm{T}_2 = \begin{bmatrix} \mathrm{R} & \boldsymbol{T} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{2.1.17}$$

which defines the relative pose of the two coordinate systems 1 and 2. The notation ${}^1\mathrm{T}_2$ signifies that the position of a homogeneous point $\mathbf{P}_2 = (p_x, p_y, p_z, 1)^T$ (given in coordinate system 2) in the reference system can be determined by multiplication with this transformation matrix: $\mathbf{P}_1 = {}^1\mathrm{T}_2\mathbf{P}_2$. Concatenating multiple transformations allows to combine them to a single one, e.g. ${}^1\mathrm{T}_4 = {}^1\mathrm{T}_2\,{}^2\mathrm{T}_3\,{}^3\mathrm{T}_4$.

2.1.4.2. Rodrigues vector $\boldsymbol{R}$

Utilizing rotation matrices to represent rotations in 3D space is convenient as a simple multiplication can be used to rotate points and vectors or transfer them from one coordinate system to another. Furthermore it is illustrative as the columns of R are the basis vectors of a coordinate system. When it comes to parameter estimation, however, having a redundant representation can be quite cumbersome. Then either more equations than

degrees of freedom are needed, or additional constraints have to be used to solve the task, which in this case are non-linear. For this reason, a minimal representation of a rotation in 3D space would be beneficial. It needs to have exactly three parameters. When using a rotation axis and a corresponding angle to represent a rotation, the number of parameters is reduced to four, which is still redundant. But as for the axis only the direction is of importance, it can be described by a 3D vector of arbitrary length. That length can actually be used to represent the angle. Explicitly, a vector $\boldsymbol{R} \in \mathbb{R}^3$ can be utilized, where

$$\theta = ||\boldsymbol{R}|| \tag{2.1.18}$$

is the rotation angle and

$$\boldsymbol{r} = \frac{\boldsymbol{R}}{\theta} \tag{2.1.19}$$

is the normalized rotation axis. $\boldsymbol{R}$ will be called *Rodrigues vector* throughout this work. It can be determined from a rotation matrix R by calculating rotation axis

$$\boldsymbol{a} = \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} \tag{2.1.20}$$

and rotation angle

$$\theta = \arctan\left(\frac{|\boldsymbol{a}|}{r_{11} + r_{22} + r_{33} - 1}\right). \tag{2.1.21}$$

With $\boldsymbol{r} = \boldsymbol{a}/|\boldsymbol{a}|$, this gives

$$\boldsymbol{R} = \boldsymbol{r}\theta. \tag{2.1.22}$$

For the inverse direction, i.e. determining the rotation matrix R from $\boldsymbol{R}$, the *Rodrigues formula* can be used:

$$\mathrm{R}(\boldsymbol{R}) = \mathrm{I}_3 + \frac{\sin\theta}{\theta}\mathrm{S}(\boldsymbol{R}) + \frac{1 - \cos\theta}{\theta^2}\mathrm{S}^2(\boldsymbol{R}) \tag{2.1.23}$$

where $\mathrm{S}(\boldsymbol{R})$ is the skew matrix constructed from $\boldsymbol{R}$.
With the help of the Taylor expansions of sine and cosine, the Rodrigues formula can be converted to

$$\mathrm{R}(\boldsymbol{R}) = \mathrm{I}_3 + \mathrm{S}(\boldsymbol{R}) + \frac{1}{2!}\mathrm{S}^2(\boldsymbol{R}) + \frac{1}{3!}\mathrm{S}^3(\boldsymbol{R}) + ... \tag{2.1.24}$$

This is also the definition of the matrix exponential. Therefore

$$\mathrm{R}(\boldsymbol{R}) = e^{\mathrm{S}(\boldsymbol{R})} \tag{2.1.25}$$

which is a useful representation when determining derivatives of functions with rotation matrices. In fact, a common operation is the transformation of a point $\boldsymbol{P}$ with a rotation matrix R, i.e.

$$\boldsymbol{f}(\boldsymbol{R}) = \mathrm{R}(\boldsymbol{R})\boldsymbol{P}. \tag{2.1.26}$$

Differentiating $\boldsymbol{f}$ with respect to $\boldsymbol{R}$ can now be done in the following way:

$$\begin{aligned}\frac{\partial \boldsymbol{f}(\boldsymbol{R})}{\partial \boldsymbol{R}} &= \frac{\partial}{\partial \boldsymbol{R}}\left(e^{\mathrm{S}(\boldsymbol{R})}\boldsymbol{P}\right) \\ &= e^{\mathrm{S}(\boldsymbol{R})}\frac{\partial}{\partial \boldsymbol{R}}\left(\mathrm{S}(\boldsymbol{R})\boldsymbol{P}\right) \\ &= \mathrm{R}(\boldsymbol{R})\frac{\partial}{\partial \boldsymbol{R}}\left(\begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}\right) \\ &= -\mathrm{R}(\boldsymbol{R})\mathrm{S}(\boldsymbol{P}).\end{aligned} \tag{2.1.27}$$

2.1.4.3. Rotation within the common plane of two vectors

The rotation matrix R_{ab} that maps a vector $\boldsymbol{a}$ to the vector $\boldsymbol{b}$ (both spherically normalized) in their common plane such as

$$\boldsymbol{b} = \mathrm{R}_{ab}\boldsymbol{a} \tag{2.1.28}$$

is given by

$$\mathrm{R}_{ab} = \mathrm{I}_d + 2\boldsymbol{b}\boldsymbol{a}^T - \frac{1}{1+\boldsymbol{a}^T\boldsymbol{b}}(\boldsymbol{a}+\boldsymbol{b})(\boldsymbol{a}+\boldsymbol{b})^T. \tag{2.1.29}$$

The dimension d of the vectors is arbitrary, the size of the identity matrix I_d has to be chosen accordingly. For further details, confer [FW14], Section 7.6.2.

2.2. Uncertainty propagation

Providing a camera model that also incorporates uncertainties is one of the main goals of this work. These uncertainties are used during the calibration procedure as well as for the final model. Also, subsequent applications can benefit from this additional information. This section therefore defines measurement uncertainties and also covers the aspect of uncertainty propagation for non-linear functions.

2.2.1. Measurement uncertainties

Any taken measurement is inevitably influenced by the inaccuracies of the observation process. Therefore, the true value can never be determined, but only a perturbed version of it. These perturbations are called *measurement uncertainties* and will generally be

considered to be normally distributed in this work. Thus every single measurement l_i is the sum of the (usually unknown) true value $\tilde{l}$ and an error value sample r_i:

$$l_i = \tilde{l} + r_i. \tag{2.2.1}$$

This r_i is assumed to be the sample of a Gaussian normal distribution $r \sim \mathsf{N}(0, \sigma_r^2)$ with zero mean and variance σ_r^2. Put differently, every measurement l is a function of its true value and the measurement noise:

$$l = f(\tilde{l}, r). \tag{2.2.2}$$

It can therefore also be considered as a random variable with a normal probability distribution

$$l \sim \mathsf{N}(\tilde{l}, \sigma_l^2) \tag{2.2.3}$$

where $\sigma_l^2 = \sigma_r^2$. The parameters of this distribution can be estimated by taking n sample measurements l_i and calculating

$$\widehat{l} = \mathsf{E}[l] = \frac{1}{n}\sum_{i=1}^{n} l_i \tag{2.2.4}$$

where $\mathsf{E}[l]$ is the expected value of l and all l_i are considered to be equally probable. $\widehat{l}$ serves as an estimate for the sample mean $\tilde{l}$ and can be used to determine

$$\widehat{\sigma}_l^2 = \mathsf{E}[(l - \widehat{l})^2] = \frac{1}{n}\sum_{i=1}^{n}(l_i - \widehat{l})^2 \tag{2.2.5}$$

as an estimation of the variance. Extending the notations to vector-valued measurements and introducing Σ_{ll} as the covariance matrix of a multivariate normal distribution gives

$$\boldsymbol{l} \sim \mathsf{N}(\tilde{\boldsymbol{l}}, \Sigma_{ll}) \tag{2.2.6}$$

$$\widehat{\boldsymbol{l}} = \mathsf{E}[\boldsymbol{l}] = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{l}_i \tag{2.2.7}$$

$$\widehat{\Sigma}_{\boldsymbol{ll}} = \mathsf{E}[(\boldsymbol{l} - \widehat{\boldsymbol{l}})(\boldsymbol{l} - \widehat{\boldsymbol{l}})^T] = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{l}_i - \widehat{\boldsymbol{l}})(\boldsymbol{l}_i - \widehat{\boldsymbol{l}})^T. \tag{2.2.8}$$

Here, n measurement vectors $\boldsymbol{l}_i$ were used to determine the estimate $\widehat{\boldsymbol{l}}$ of the sample mean $\tilde{\boldsymbol{l}}$ and the estimated covariance matrix $\widehat{\Sigma}_{\boldsymbol{ll}}$.
If uncertain measurements $\boldsymbol{l}$ are parameters of a function, the outcome $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{l})$ is also uncertain. In this work, the corresponding uncertainty $\Sigma_{\boldsymbol{yy}}$ will be determined by either using empirical Monte Carlo simulation, analytical first order error propagation, or the unscented transform. Each method will be described in further detail in the following sections.

2.2.2. Monte Carlo simulation

Monte Carlo simulation can be used to determine the probability distribution of the output $\boldsymbol{y}$ of a (non-linear) function $\boldsymbol{f}(\boldsymbol{l})$ where the input value $\boldsymbol{l}$ is uncertain with a known distribution. The principle is to draw multiple samples $\boldsymbol{l}_i$, $i = 1..n$, from the error probability distribution (always Gaussian in this work) and to determine the corresponding outputs $\boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{l}_i)$. These are analyzed using (2.2.7) and (2.2.8) to calculate the mean value and the covariance matrix which characterize the distribution of the output values. An advantage of this method is its simplicity and its applicability to arbitrary functions. A major disadvantage, though, is its computational load. Depending on the complexity of $\boldsymbol{f}$, it might become unfeasible to determine the amount of output values necessary to get a decent approximation of their distribution.

2.2.3. First order uncertainty propagation

Although Monte Carlo simulation is always a valid means to determine the output uncertainty of a function, there are also occasions when an analytical solution is needed. Especially when the development of uncertainties within a chain of functions is to be analyzed or simply in case the necessary computational resources are not available.
For functions which are linear combinations of independent random variables $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ as in

$$\boldsymbol{y} = a\boldsymbol{l}_1 + b\boldsymbol{l}_2, \quad \boldsymbol{l}_1 \sim \mathsf{N}(\bar{\boldsymbol{l}}_1, \Sigma_{\boldsymbol{l}_1\boldsymbol{l}_1}), \quad \boldsymbol{l}_2 \sim \mathsf{N}(\bar{\boldsymbol{l}}_2, \Sigma_{\boldsymbol{l}_2\boldsymbol{l}_2})$$

the resulting covariance can be determined by using (2.2.7) and (2.2.8):

$$\begin{aligned}
\bar{\boldsymbol{y}} &= \mathsf{E}[a\boldsymbol{l}_1 + b\boldsymbol{l}_2] \\
&= a\mathsf{E}[\boldsymbol{l}_1] + b\mathsf{E}[\boldsymbol{l}_2] \\
\Rightarrow\ \bar{\boldsymbol{y}} &= a\bar{\boldsymbol{l}}_1 + b\bar{\boldsymbol{l}}_2 \qquad (2.2.9) \\
\Sigma_{\boldsymbol{y}\boldsymbol{y}} &= \mathsf{E}[(\boldsymbol{y} - \bar{\boldsymbol{y}})(\boldsymbol{y} - \bar{\boldsymbol{y}})^T] \\
&= \mathsf{E}[(a\boldsymbol{l}_1 + b\boldsymbol{l}_2 - a\bar{\boldsymbol{l}}_1 - b\bar{\boldsymbol{l}}_2)(a\boldsymbol{l}_1 + b\boldsymbol{l}_2 - a\bar{\boldsymbol{l}}_1 - b\bar{\boldsymbol{l}}_2)^T] \\
&= \mathsf{E}[a^2(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)^T + b^2(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)^T \\
&\qquad + ab(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)^T + ab(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)^T] \\
&= a^2\mathsf{E}[(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)^T] + b^2\mathsf{E}[(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)^T] \\
&\qquad + ab\underbrace{\mathsf{E}[(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)^T]}_{=0\ (\boldsymbol{l}_1 \text{ and } \boldsymbol{l}_2 \text{ indep.})} + ab\underbrace{\mathsf{E}[(\boldsymbol{l}_2 - \bar{\boldsymbol{l}}_2)(\boldsymbol{l}_1 - \bar{\boldsymbol{l}}_1)^T]}_{=0} \\
\Rightarrow\ \Sigma_{\boldsymbol{y}\boldsymbol{y}} &= a^2\Sigma_{l_1l_1} + b^2\Sigma_{l_2l_2}. \qquad (2.2.10)
\end{aligned}$$

In case the function is nonlinear, uncertainty propagation can be achieved by linearizing via first order Taylor expansion around the expected value $\bar{\boldsymbol{l}}$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{l}) = \boldsymbol{f}(\bar{\boldsymbol{l}} + \Delta\boldsymbol{l}) = \boldsymbol{f}(\bar{\boldsymbol{l}}) + \nabla\boldsymbol{f}(\bar{\boldsymbol{l}})\Delta\boldsymbol{l} + \mathsf{O}(||\Delta\boldsymbol{l}||^2). \tag{2.2.11}$$

Here, ∇ is the nabla operator which represents the first (partial) derivatives. Neglecting terms of higher order and assuming that $\bar{\boldsymbol{y}} = \boldsymbol{f}(\bar{\boldsymbol{l}})$, the covariance of $\boldsymbol{y}$ can be determined via

$$\begin{aligned}
\Sigma_{\boldsymbol{yy}} &= \mathsf{E}[(\boldsymbol{f}(\bar{\boldsymbol{l}} + \Delta\boldsymbol{l}) - \bar{\boldsymbol{y}})(\boldsymbol{f}(\bar{\boldsymbol{l}} + \Delta\boldsymbol{l}) - \bar{\boldsymbol{y}})^T] \\
&\approx \mathsf{E}[(\boldsymbol{f}(\bar{\boldsymbol{l}} + \Delta\boldsymbol{l}) - \boldsymbol{f}(\bar{\boldsymbol{l}}))(\boldsymbol{f}(\bar{\boldsymbol{l}} + \Delta\boldsymbol{l}) - \boldsymbol{f}(\bar{\boldsymbol{l}}))^T] \\
&\approx \mathsf{E}[\nabla\boldsymbol{f}(\bar{\boldsymbol{l}})\Delta\boldsymbol{l}\Delta\boldsymbol{l}^T\nabla\boldsymbol{f}(\bar{\boldsymbol{l}})^T] \\
\Rightarrow\ \Sigma_{\boldsymbol{yy}} &\approx \nabla\boldsymbol{f}(\bar{\boldsymbol{l}})\Sigma_{ll}\nabla\boldsymbol{f}(\bar{\boldsymbol{l}})^T.
\end{aligned} \tag{2.2.12}$$

When working in projective spaces, many times implicit relations arise. Therefore, it might be difficult to get an explicit realization with respect to a specific variable. Nevertheless, determining the corresponding covariance is still straight forward. Given the implicit function

$$\boldsymbol{g}(\boldsymbol{l}_1, \boldsymbol{l}_2) = \boldsymbol{0} \tag{2.2.13}$$

it is possible to identify the derivative of $\boldsymbol{l}_2$ with respect to $\boldsymbol{l}_1$ by using the total derivative:

$$\begin{aligned}
& \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_1}\mathrm{d}\boldsymbol{l}_1 + \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_2}\mathrm{d}\boldsymbol{l}_2 = \boldsymbol{0} \\
\Leftrightarrow \quad & \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_1} + \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_2}\frac{\mathrm{d}\boldsymbol{l}_2}{\mathrm{d}\boldsymbol{l}_1} = \boldsymbol{0} \\
\Leftrightarrow \quad & \frac{\mathrm{d}\boldsymbol{l}_2}{\mathrm{d}\boldsymbol{l}_1} = -\left(\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_2}\right)^{-1}\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_1} \\
\Leftrightarrow \quad & \frac{\mathrm{d}\boldsymbol{l}_2}{\mathrm{d}\boldsymbol{l}_1} = -\mathrm{B}^{-1}\mathrm{A} \quad \text{with} \quad \mathrm{B} = \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_2},\ \mathrm{A} = \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{l}_1}.
\end{aligned} \tag{2.2.14}$$

Utilizing (2.2.12), the covariances of $\boldsymbol{l}_2$ can be determined with the help of the covariance matrix $\Sigma_{\boldsymbol{l}_1\boldsymbol{l}_1}$ of $\boldsymbol{l}_1$, even though no explicit formulation of $\boldsymbol{l}_2$ as a function of $\boldsymbol{l}_1$ has been derived:

$$\begin{aligned}
\Sigma_{\boldsymbol{l}_2\boldsymbol{l}_2} &= \nabla\boldsymbol{l}_2\Sigma_{\boldsymbol{l}_1\boldsymbol{l}_1}\nabla\boldsymbol{l}_2^T \\
&= \frac{\mathrm{d}\boldsymbol{l}_2}{\mathrm{d}\boldsymbol{l}_1}\Sigma_{\boldsymbol{l}_1\boldsymbol{l}_1}\left(\frac{\mathrm{d}\boldsymbol{l}_2}{\mathrm{d}\boldsymbol{l}_1}\right)^T \\
\Rightarrow \quad \Sigma_{\boldsymbol{l}_2\boldsymbol{l}_2} &\overset{(2.2.14)}{=} \mathrm{B}^{-1}\mathrm{A}\Sigma_{\boldsymbol{l}_1\boldsymbol{l}_1}\mathrm{A}^T\mathrm{B}^{-T}.
\end{aligned} \tag{2.2.15}$$

2.2.3.1. Example 1: Uncertain homogeneous transformation of an uncertain point

A homogeneous transformation ${}^1\mathrm{T}_2$ consists of a rotation matrix R and a translation vector $\boldsymbol{t}$:

$$ {}^1\mathrm{T}_2 = \begin{bmatrix} \mathrm{R} & \boldsymbol{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix}. \tag{2.2.16} $$

Both elements are assumed to be random variables with a Gaussian probability distribution:

$$ \mathrm{R} \sim \mathsf{R}(\mathsf{N}(\bar{\boldsymbol{R}}, \Sigma_{\boldsymbol{RR}})) \quad \text{and} \quad \boldsymbol{t} \sim \mathsf{N}(\bar{\boldsymbol{t}}, \Sigma_{\boldsymbol{t}}). \tag{2.2.17} $$

$\bar{\boldsymbol{R}}$ is in this case the mean 3D Rodrigues vector representing the rotation (confer Section 2.1.4.2). Multiplying ${}^1\mathrm{T}_2$ with an uncertain homogeneous point $\mathbf{P}_2 \sim \mathsf{N}(\bar{\mathbf{P}}_2, \Sigma_{\mathbf{P}_2\mathbf{P}_2})$, defined in coordinate system 2, gives the coordinates of that point in system 1:

$$ \mathbf{P}_1 = {}^1\mathrm{T}_2 \, \mathbf{P}_2. \tag{2.2.18} $$

This equation can also be written in an inhomogeneous form as

$$ \boldsymbol{P}_1 = \mathrm{R} \, \boldsymbol{P}_2 + \boldsymbol{t}. \tag{2.2.19} $$

Assuming R, $\boldsymbol{t}$ and $\boldsymbol{P}_2$ to be mutually independent, the standard procedure for uncertainty propagation (Section 2.2.3) can be applied to get the covariance matrix of $\boldsymbol{P}_1$:

$$ \Sigma_{\boldsymbol{P}_1\boldsymbol{P}_1} = \mathrm{J}_p \Sigma_{\boldsymbol{p}_2\boldsymbol{p}_2} \mathrm{J}_p^T + \mathrm{J}_t \Sigma_{\boldsymbol{tt}} \mathrm{J}_t^T + \mathrm{J}_R \Sigma_{\boldsymbol{RR}} \mathrm{J}_R^T. \tag{2.2.20} $$

Determining the Jacobians as

$$ \begin{aligned} \mathrm{J}_p &= \frac{\partial \boldsymbol{P}_1}{\partial \boldsymbol{P}_2} = \mathrm{R} \\ \mathrm{J}_t &= \frac{\partial \boldsymbol{P}_1}{\partial \boldsymbol{t}} = \mathrm{I}_3 \\ \mathrm{J}_R &= \frac{\partial \boldsymbol{P}_1}{\partial \boldsymbol{R}} = -\mathrm{R} \cdot \mathsf{S}(\boldsymbol{P}_2) \end{aligned} $$

finally delivers

$$ \Sigma_{\boldsymbol{P}_1\boldsymbol{P}_1} = \mathrm{R} \, \Sigma_{\boldsymbol{P}_2\boldsymbol{P}_2} \mathrm{R}^T + \Sigma_{\boldsymbol{tt}} + \mathrm{R} \, \mathsf{S}(\boldsymbol{P}_2) \Sigma_{\boldsymbol{RR}} \mathsf{S}^T(\boldsymbol{P}_2) \mathrm{R}^T. \tag{2.2.21} $$

2.2.3.2. Example 2: Uncertain Rodrigues vector from uncertain rotation matrix

Given are a rotation matrix R with the parameters $\{r_{11}, r_{21}, ..., r_{33}\}$ and the corresponding (9×9) covariance matrix Σ_{RR} of all elements. As defined in (2.1.22), the Rodrigues vector $\boldsymbol{R}$ is obtained via

$$ \boldsymbol{R} = \boldsymbol{r}\theta $$

with

$$\boldsymbol{r} = \frac{\boldsymbol{a}}{|\boldsymbol{a}|}, \quad \boldsymbol{a} = \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}, \quad \theta = \arctan\left(\frac{|\boldsymbol{a}|}{d_t}\right), \quad d_t = r_{11} + r_{22} + r_{33} - 1.$$

To determine the covariance matrix $\Sigma_{\boldsymbol{RR}}$ of the Rodrigues vector, partial derivatives for all parameters are needed to form the Jacobian

$$\mathrm{J}_R = \begin{bmatrix} \frac{\partial R_1}{\partial r_{11}} & 0 & 0 & 0 & \frac{\partial R_1}{\partial r_{22}} & \frac{\partial R_1}{\partial r_{32}} & 0 & \frac{\partial R_1}{\partial r_{23}} & \frac{\partial R_1}{\partial r_{33}} \\ \frac{\partial R_2}{\partial r_{11}} & 0 & \frac{\partial R_2}{\partial r_{31}} & 0 & \frac{\partial R_2}{\partial r_{22}} & 0 & \frac{\partial R_2}{\partial r_{13}} & 0 & \frac{\partial R_2}{\partial r_{22}} \\ \frac{\partial R_3}{\partial r_{11}} & \frac{\partial R_3}{\partial r_{21}} & 0 & \frac{\partial R_3}{\partial r_{12}} & \frac{\partial R_3}{\partial r_{22}} & 0 & 0 & 0 & \frac{\partial R_3}{\partial r_{33}} \end{bmatrix}. \tag{2.2.22}$$

In further detail, they are

$$\frac{\partial R_i}{\partial r_{jk}} = \frac{\partial r_i}{\partial r_{jk}} \cdot \theta + r_i \cdot \frac{\partial \theta}{\partial r_{jk}} \tag{2.2.23}$$

with

$$\begin{aligned}
&\frac{\partial r_i}{\partial r_{jk}} = 0 \quad \text{for } j = k \\
&\frac{\partial \theta}{\partial r_{11}} = \frac{\partial \theta}{\partial r_{22}} = \frac{\partial \theta}{\partial r_{33}} = -\frac{|\boldsymbol{a}|}{d_t^2 + 1} \\
&\frac{\partial r_1}{\partial r_{32}} = -\frac{\partial r_1}{\partial r_{23}} = -\frac{a_1^2}{|\boldsymbol{a}|^3} + \frac{1}{|\boldsymbol{a}|}, \quad \frac{\partial \theta}{\partial r_{32}} = -\frac{\partial \theta}{\partial r_{23}} = \frac{a_1}{|\boldsymbol{a}|(d_t + \frac{1}{d_t})} \\
&\frac{\partial r_2}{\partial r_{13}} = -\frac{\partial r_2}{\partial r_{31}} = -\frac{a_2^2}{|\boldsymbol{a}|^3} + \frac{1}{|\boldsymbol{a}|}, \quad \frac{\partial \theta}{\partial r_{13}} = -\frac{\partial \theta}{\partial r_{31}} = \frac{a_2}{|\boldsymbol{a}|(d_t + \frac{1}{d_t})} \\
&\frac{\partial r_3}{\partial r_{21}} = -\frac{\partial r_3}{\partial r_{12}} = -\frac{a_3^2}{|\boldsymbol{a}|^3} + \frac{1}{|\boldsymbol{a}|}, \quad \frac{\partial \theta}{\partial r_{21}} = -\frac{\partial \theta}{\partial r_{12}} = \frac{a_3}{|\boldsymbol{a}|(d_t + \frac{1}{d_t})}.
\end{aligned}$$

This allows to calculate

$$\Sigma_{\boldsymbol{RR}} = \mathrm{J}_R \Sigma_{\mathrm{RR}} \mathrm{J}_R^T. \tag{2.2.24}$$

2.2.4. Unscented transform

The previous sections introduced two methods for uncertainty propagation. They both have their merits, but also suffer from severe drawbacks. While Monte Carlo estimation is computationally expensive, first order propagation might not be adequate due to the non-linear character of a function. The *unscented transform* tries to avoid these pitfalls. It transforms samples of the assumed probability distribution of the input values to the output domain by using the exact function and is in this respect similar to the Monte

Carlo method. But instead of doing this as often as deemed necessary to get a decent approximation of the output distribution, the samples are drawn according to special rules which allows to get a good approximation from significantly less data points.
The idea was first proposed by Julier and Uhlmann in [JU97] in the context of Kalman filtering. They call the utilized samples *sigma points* $\mathcal{L}$. For an input value $\boldsymbol{l}$ of dimension d they determine them in the following way:

$$\mathcal{L}_0 = \bar{\boldsymbol{l}} \tag{2.2.25}$$

$$\mathcal{L}_i = \bar{\boldsymbol{l}} + \left[\sqrt{(d+\kappa)\Sigma_{ll}}\right]_i \tag{2.2.26}$$

$$\mathcal{L}_{i+d} = \bar{\boldsymbol{l}} - \left[\sqrt{(d+\kappa)\Sigma_{ll}}\right]_i . \tag{2.2.27}$$

Here $[\sqrt{\Sigma_{ll}}]_i$ stands for the ith column of the corresponding matrix. The necessary square root can be determined with the help of the Cholesky decomposition. In case the probability distribution of $\boldsymbol{l}$ is Gaussian, it is proposed to choose $\kappa = d - 3$. These sigma points are now mapped to the output space to form the output values

$$\mathcal{Y}_i = \boldsymbol{f}(\mathcal{L}_i). \tag{2.2.28}$$

Julier and Uhlmann furthermore assign the weights

$$\mathcal{W}_0 = \frac{\kappa}{d+\kappa} \tag{2.2.29}$$

$$\mathcal{W}_i = \frac{1}{2(d+\kappa)}, \quad i = 1..d \tag{2.2.30}$$

$$\mathcal{W}_{i+d} = \frac{1}{2(d+\kappa)}, \quad i = 1..d \tag{2.2.31}$$

which serve to determine the mean output value

$$\bar{\boldsymbol{y}} = \sum_{i=0}^{2d} \mathcal{W}_i \mathcal{Y}_i. \tag{2.2.32}$$

The desired corresponding covariance matrix Σ_{yy} is finally calculated as

$$\Sigma_{\boldsymbol{yy}} = \sum_{i=0}^{2d} \mathcal{W}_i (\mathcal{Y}_i - \bar{y})(\mathcal{Y}_i - \bar{y})^T. \tag{2.2.33}$$

2.3. B-splines

One of the main aspects in this work is the generation of continuous descriptions of mappings which are only known from sparse sample points. They serve as approximations of

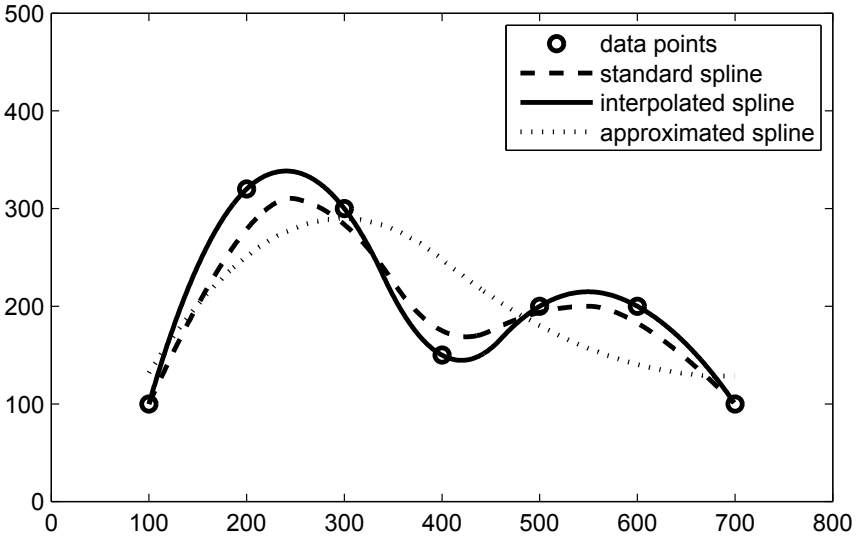


Figure 2.3.1.: Spline curves through given data points. The different methods of spline construction *standard* (data points are directly used as control points), *interpolation* and *approximation* are described in Sections 2.3, 2.3.2 and 2.3.3, respectively. Please note that the actual control points $\boldsymbol{P}_i$ are not shown here.

the real unknown mappings and are a means for interpolating between the samples. B-splines are chosen for this task because they offer a great flexibility and therefore have the ability to cope with many situations that arise in this work. The current section focuses on those aspects of B-splines that are relevant for the remainder of this work. For more information, detailed derivations and proofs and also computationally efficient implementations the reader is referred to "The *NURBS* Book" written by Piegl and Tiller [PT97]. Figure 2.3.1 shows a simple example with the results of the three construction methods *standard*, *interpolation* and *approximation* which will be introduced in the course of this section.

Splines are polynomial functions that are defined piecewise, i.e. different sections in its parameter domain have different representations. The parameter for a spline curve is called u in this work. Its domain can be selected arbitrarily, but will be $[0, 1]$ here. It is divided into sections called *spans* by placing $n_k + 1$ *knots* $u_0 ... u_{n_k}$ within this interval where $u_0 = 0$, $u_{n_k} = 1$ and $u_i \leq u_{i+1}$. All spans s_i have the range $[u_i, u_{i+1})$, except for s_{n_k-1} which is defined over the closed interval $[u_{n_k-1}, u_{n_k}]$. It is explicitly allowed for two subsequent knots u_i and u_{i+1} to have the same value. The corresponding span then has length 0 and the continuity degree of the spline is reduced. A B-spline curve $\boldsymbol{C}(u)$ consists of basis functions $N_{i,p}(u)$, $i \in [0, n]$, which are polynomials of degree p and are defined

recursively as

$$\begin{aligned} N_{i,p}(u) &= \frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_i}N_{i+1,p-1}(u) \\ \text{with} \quad N_{i,0}(u) &= \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.3.1}$$

These functions are only dependent on the selected knots u_i. The course of a B-spline curve is dictated by control points $\boldsymbol{P}_i$, which can be considered as *attractors* of the curve. This leads to the final B-spline function

$$\boldsymbol{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P}_i \tag{2.3.2}$$

where the dimension d of the curve $\boldsymbol{C}(u)$ is defined by the dimension of the control points. The smoothness of the curve is mainly influenced by the degree p of the spline. When it is set to 1, the resulting curve linearly connects all the control points. Then only $p+1=2$ control points influence the course of the curve within each span. Increasing the degree increases the number of control points that contribute to each section and therefore leads to a smoother curve, but stops the spline from actually passing through the control points. A means to force the curve to pass through a control point is to increase the multiplicity of a knot. This decreases the continuity degree, i.e. the number of continuous derivatives, at that point. In this work, multiple knots will only be used to ensure that a spline function starts exactly at the first control point and ends at the last. Therefore, given $n+1$ control points $\{\boldsymbol{P}_0 ... \boldsymbol{P}_n\}$ and a desired degree of p, the knot vector will be

$$U = \{\underbrace{0, ..., 0}_{p+1}, u_{p+1}, ..., u_n, \underbrace{1, ..., 1}_{p+1}\}. \tag{2.3.3}$$

The reader may confer Chapter 2.2 of [PT97] for further details on knot multiplicity and spline continuity.
In contrast to spline curves, which are one-dimensional functions in d-dimensional space, *spline surfaces* are two-dimensional mappings to d-dimensional space. They depend on two parameters, called u and v, both defined over $[0,1]$, and are partitioned by knots $\{u_0, .., u_{n_k}\}$ and $\{v_0, .., v_{m_k}\}$ in the two directions which define an $(n_k \times m_k)$ grid of span patches. Control points now lie on a $(n+1 \times m+1)$ grid and influence the shape of the surface in a similar way as for curves. The final function of a B-spline surface with degree p reads as

$$\boldsymbol{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,p}(v)\boldsymbol{P}_{ij} \tag{2.3.4}$$

where the $N_{j,p}(v)$ depend on the knots v_j and are defined by recursive equations equivalent to (2.3.1).

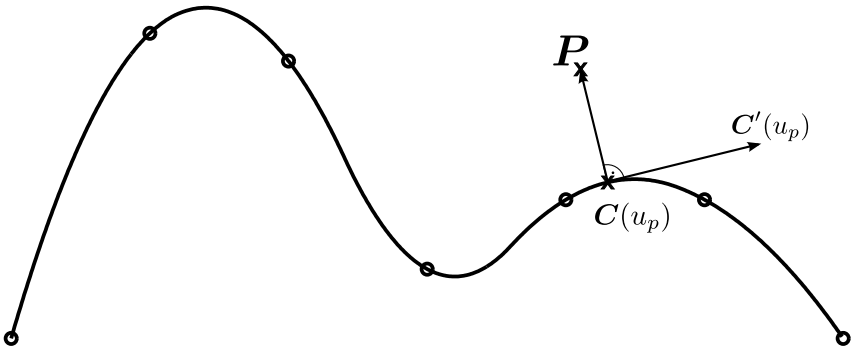


Figure 2.3.2.: *Spline projection* is the process of identifying the parameter u_p for which the corresponding curve point $\boldsymbol{C}(u_p)$ lies closest to a point $\boldsymbol{P}$, i.e. where the tangent vector $\boldsymbol{C}'(u_p)$ is perpendicular to the vector from $\boldsymbol{C}(u_p)$ to $\boldsymbol{P}$. The same procedure will be used when the point lies exactly on the curve and is then called *spline inversion*.

2.3.1. Inversion of B-splines

Spline inversion is the process of identifying the parameter value u_p for which a curve passes through a specified point $\boldsymbol{P}$, i.e. where $\boldsymbol{C}(u_p) = \boldsymbol{P}$ (or finding u_p and v_p which fulfill $\boldsymbol{S}(u_p, v_p) = \boldsymbol{P}$ in case of a surface). The most obvious approach would start with identifying the span in which $\boldsymbol{P}$ lies. The next step would then be to determine the coefficients of the corresponding basis function and inverting the corresponding d equations (one for each dimension of $\boldsymbol{P}$) separately to find a common solution. But this is computationally challenging for degrees bigger than 3 and might give different solutions for each dimension, especially when $\boldsymbol{P}$ does not lie exactly on the spline.
The approach chosen here is much more general, as it is independent of the degree of the spline and also works for points that do not lie on the spline. In the latter case the procedure is called *spline projection* and the parameter of the curve point closest to $\boldsymbol{P}$ will be determined. The principal idea is that the desired parameter value u_p is found when the derivative $\boldsymbol{C}'(u_p)$ and the vector from the given point $\boldsymbol{P}$ to the point on the spline $\boldsymbol{C}(u_p)$ are perpendicular (as illustrated in Fig. 2.3.2). This gives the relationship

$$f(u) = (\boldsymbol{C}(u) - \boldsymbol{P})^T \boldsymbol{C}'(u) = 0. \tag{2.3.5}$$

The corresponding task is to find

$$u_p = \underset{u}{\operatorname{argmin}} \, (\boldsymbol{C}(u) - \boldsymbol{P})^T \boldsymbol{C}'(u) \tag{2.3.6}$$

which can be solved by utilizing Newton's algorithm for root finding. The kth derivative of a B-spline basis function (given here without proof, confer [PT97], pp. 59 for details) is calculated via

$$N_{i,p}^{(k)}(u) = p \left(\frac{N_{i,p-1}^{(k-1)}(u)}{u_{i+p} - u_i} + \frac{N_{i+1,p-1}^{(k-1)}(u)}{u_{i+p+1} - u_i} \right) \tag{2.3.7}$$

which leads to the curve derivative

$$\boldsymbol{C}^{(k)}(u) = \sum_{i=0}^{n} N_{i,p}^{(k)}(u)\boldsymbol{P}_i. \tag{2.3.8}$$

To determine an initial value u_t for the root finding process, the following procedure is used:

1. Determine all span midpoints $\boldsymbol{C}_{im} = \boldsymbol{C}(u_{im})$ with $u_{im} = (u_i + u_{i+1})/2$.

2. Identify that $\boldsymbol{C}_{im}$ which has the smallest distance to $\boldsymbol{P}$ and use the corresponding u_{im} as starting point u_t.

The Newton iteration is now

$$u_{t+1} = u_t - \frac{f(u_t)}{f'(u_t)} = u_t - \frac{(\boldsymbol{C}(u_t) - \boldsymbol{P})^T\boldsymbol{C}'(u_t)}{\boldsymbol{C}'^T(u_t)\boldsymbol{C}'(u_t) + (\boldsymbol{C}(u_t) - \boldsymbol{P})^T\boldsymbol{C}''(u_t)}. \tag{2.3.9}$$

As a stopping criterion

$$a = \frac{|(\boldsymbol{C}(u_t) - \boldsymbol{P})^T\boldsymbol{C}'(u_t)|}{|(\boldsymbol{C}(u_t) - \boldsymbol{P})||\boldsymbol{C}'(u_t)|} \tag{2.3.10}$$

has to drop below a specified threshold, which signifies that the two vectors are close to orthogonal.
For spline surfaces $\boldsymbol{S}(u, v)$, the procedure works in the same way. Now there are two functions

$$f_s(u, v) = (\boldsymbol{S}(u, v) - \boldsymbol{P})^T\boldsymbol{S}_u(u, v) \tag{2.3.11}$$

$$g_s(u, v) = (\boldsymbol{S}(u, v) - \boldsymbol{P})^T\boldsymbol{S}_v(u, v) \tag{2.3.12}$$

which both need to be 0. $\boldsymbol{S}_u$ and $\boldsymbol{S}_v$ are the first partial derivatives in u and v direction, respectively. Generally the surface derivatives are defined as

$$\frac{\partial^{k+l}}{\partial^k u \partial^l v}\boldsymbol{S}(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}^{(k)}(u)N_{j,p}^{(l)}(v)\boldsymbol{P}_{ij} \tag{2.3.13}$$

with basis function derivatives $N_{i,p}^{(k)}$ and $N_{j,p}^{(l)}$ given by (2.3.7) (compare [PT97], p.111).
After determining starting values u_t and v_t for the root finding procedure similarly as before, the Jacobian

$$J_t = \left[\begin{array}{cc} f_{s_u} & f_{s_v} \\ g_{s_u} & g_{s_v} \end{array}\right]\Bigg|_{\substack{u=u_t \\ v=v_t}} \tag{2.3.14}$$

$$= \left[\begin{array}{cc} \boldsymbol{S}_u^T\boldsymbol{S}_u + (\boldsymbol{S} - \boldsymbol{P})^T\boldsymbol{S}_{uu} & \boldsymbol{S}_v^T\boldsymbol{S}_u + (\boldsymbol{S} - \boldsymbol{P})^T\boldsymbol{S}_{uv} \\ \boldsymbol{S}_u^T\boldsymbol{S}_v + (\boldsymbol{S} - \boldsymbol{P})^T\boldsymbol{S}_{vu} & \boldsymbol{S}_v^T\boldsymbol{S}_v + (\boldsymbol{S} - \boldsymbol{P})^T\boldsymbol{S}_{vv} \end{array}\right]\Bigg|_{\substack{u=u_t \\ v=v_t}} \tag{2.3.15}$$

serves to formulate the equation system

$$J_t \delta_t = \kappa_t \tag{2.3.16}$$

with

$$\delta_t = \begin{pmatrix} u_{t+1} - u_t \\ v_{t+1} - v_t \end{pmatrix} \quad \text{and} \quad \kappa_t = - \begin{pmatrix} f_s(u_t, v_t) \\ g_s(u_t, v_t) \end{pmatrix}. \tag{2.3.17}$$

By solving (2.3.16), u_{t+1} and v_{t+1} are determined. Convergence is reached when a_1 and a_2 defined as

$$a_1 = \frac{|(\boldsymbol{S} - \boldsymbol{P})^T \boldsymbol{S}_u|}{|(\boldsymbol{S} - \boldsymbol{P})||\boldsymbol{S}_u|}\bigg|_{\substack{u=u_t \\ v=v_t}} \quad \text{and} \quad a_1 = \frac{|(\boldsymbol{S} - \boldsymbol{P})^T \boldsymbol{S}_v|}{|(\boldsymbol{S} - \boldsymbol{P})||\boldsymbol{S}_v|}\bigg|_{\substack{u=u_t \\ v=v_t}} \tag{2.3.18}$$

become small.

2.3.2. B-spline interpolation

In the last sections, control points were used to influence the course of a spline curve or the structure of a surface. Setting and modifying them directly is a common procedure in curve and surface design, where the main goal is to obtain "nicely shaped" functions.
Another application of splines is to get continuous descriptions for mappings that are sampled at discrete positions. Then, the spline should pass through the obtained data points and serve as an interpolation function that allows to calculate values also for intermediate positions, where no data has been gathered. Therefore, the goal is now to determine the control points $\boldsymbol{P}_i$ such as the spline runs through or at least passes closely by the data points $\boldsymbol{Q}_k$, $k \in [0, n_{qu}]$. This can be achieved by solving a linear system made of $n+1$ equations where the control points $\boldsymbol{P}_i$ are the unknowns:

$$\boldsymbol{Q}_k = \boldsymbol{C}(\bar{u}_k) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k) \boldsymbol{P}_i. \tag{2.3.19}$$

To be able to do so, the parameter values $\bar{u}_k$ have to be chosen. They define the parameter position along the spline for each data point. Furthermore, a knot vector $\{u_0, ..., u_{n_k}\}$ has to be constructed. There are several methods to select parameters $\bar{u}_k$ and knots u_i ([PT97], pp 364). In this work *uniform parameterization* will be used exclusively. This means that parameters and knots (except for the multiple ones at the beginning and at the end) are equally spaced:

$$\bar{u}_k = \frac{k}{n} \quad \text{for} \quad k = 0, ..., n \tag{2.3.20}$$

and

$$\begin{aligned} u_0 = ... = u_p = 0 \qquad & u_{n+1} = ... = u_{n_k} = 1 \\ u_{j+p} = \frac{j}{n-p+1} \quad & \text{for} \quad j = 1, ..., n-p. \end{aligned} \tag{2.3.21}$$

Now (2.3.19) can be utilized to stack the equations from the $n+1$ data points and set up a $(n+1 \times n+1)$ matrix A_C which contains the coefficients of the basis functions at parameter values $\bar{u}_k$. This leads to the equation system

$$\underset{(n+1\times n+1)}{\mathrm{A}_C} \underset{(n+1\times d)}{\mathrm{P}} = \underset{(n+1\times d)}{\mathrm{Q}} \tag{2.3.22}$$

where P and Q contain the stacked control points and data points, respectively, as row vectors. Inverting A_C and calculating

$$\mathrm{P} = \mathrm{A}_C^{-1}\mathrm{Q} \tag{2.3.23}$$

delivers the desired control points $\boldsymbol{P}_i$. This procedure is called *spline interpolation*, because the resulting spline passes exactly through the data points.
For *surface interpolation*, the control points $\boldsymbol{P}_{ij}$ have to be determined such as the spline surface passes through a grid of data points $\boldsymbol{Q}_{kl}$. This signifies that for selected parameter values $(\bar{u}_k, \bar{v}_l)$ the equation

$$\boldsymbol{Q}_{kl} = \boldsymbol{S}(\bar{u}_k, \bar{v}_l) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\bar{u}_k)N_{j,p}(\bar{v}_l)\boldsymbol{P}_{ij} \tag{2.3.24}$$

has to be fulfilled. The $\bar{u}_k$ and $\bar{v}_l$ are determined as proposed in (2.3.20) and the knots u_i and v_i as shown in (2.3.21). As there are $(n+1)(m+1)$ equations like (2.3.24), a square matrix A_S with that amount of rows and columns could be constructed and used to determine the control points. To avoid the computationally intensive inversion of this huge matrix, (2.3.24) can be rewritten as

$$\boldsymbol{Q}_{kl} = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\left(\sum_{j=0}^{m} N_{j,p}(\bar{v}_l)\boldsymbol{P}_{ij}\right) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\boldsymbol{R}_{il} \tag{2.3.25}$$

with

$$\boldsymbol{R}_{il} = \sum_{j=0}^{m} N_{j,p}(\bar{v}_l)\boldsymbol{P}_{ij}. \tag{2.3.26}$$

This allows to first determine all $\boldsymbol{R}_{il}$ as if they contained control points of spline curves that pass through data points $\boldsymbol{Q}_{0l}, ..., \boldsymbol{Q}_{nl}$ with parameters $\bar{u}_k$ and knots u_i. These can then be used to calculate the corresponding $\boldsymbol{P}_{il}$ by using (2.3.26) with the $\bar{v}_l$ and the v_j. Doing this for all $l = 0, ..., m$ gives the complete set of control points. This method dramatically reduces the computational complexity compared to the direct approach.

2.3.3. B-spline approximation

In case each data point is subject to noise, interpolation as described before would not be appropriate. In fact it is beneficial to determine the control points of a spline such

as it does not follow the measurement errors, but is a smooth approximation of the actual measurements. This reduces the influence of the noise. The result is a continuous description that supposedly is a better representation of the underlying true mapping. This procedure is called *curve approximation* or *surface approximation* and generally resembles the interpolation process. A major difference is that now more data points than control points are used, which leads to the system (2.3.22) being overdetermined. Having data points $\boldsymbol{Q}_0, ..., \boldsymbol{Q}_{n_{qu}}$, the corresponding parameter values $\bar{u}_k$ are calculated with uniform distance as before:

$$\bar{u}_k = \frac{k}{n_{qu}} \quad \text{for} \quad k = 0, ..., n_{qu}. \tag{2.3.27}$$

The curve is still defined by $n + 1$ control points $\boldsymbol{P}_i$, where $n < n_{qu}$, with knots at positions given by (2.3.21). Determining the $\boldsymbol{P}_i$ is equivalent to minimizing the sum of squared differences

$$\sum_{k=0}^{n_{qu}} |\boldsymbol{Q}_k - \boldsymbol{C}(\bar{u}_k)|^2 \tag{2.3.28}$$

which can be achieved by the ordinary least squares approach:

$$\mathrm{P} = (\mathrm{A}_C^T \mathrm{A}_C)^{-1} \mathrm{A}_C^T \mathrm{Q}. \tag{2.3.29}$$

This concept can be applied to spline surfaces as well: Given data points on a $(n_{qu}+1 \times n_{qv}+1)$ grid, control points $\boldsymbol{P}_{00}, ..., \boldsymbol{P}_{nm}$ with $n < n_{qu}$ and $m < n_{qv}$ can be determined using the procedure described at the end of Section 2.3.2. Now, uniformly spaced parameter values $\bar{v}_0, ..., \bar{v}_{n_{qv}}$ serve to first calculate the $\boldsymbol{R}_{il}$ in a least squares sense and subsequently the desired $\boldsymbol{P}_{ij}$.

2.4. Estimation

The goal of *parameter estimation* is to determine the parameters $\boldsymbol{x}$ of a system described by a model $\boldsymbol{f}(\boldsymbol{x})$. The model is a mathematical description of a process which can for example be derived from physical laws or geometric relations. Commonly, there is no direct access to the model parameters. Instead, the output $\boldsymbol{y}$ of such a system is available and can be assessed by taking *measurements* or *observations* $\boldsymbol{l}$ (both terms will be used as synonyms in this work). With a perfect model and noiseless measurements

$$\tilde{\boldsymbol{l}} = \boldsymbol{y} = \boldsymbol{f}(\tilde{\boldsymbol{x}}) \tag{2.4.1}$$

where the tilde marks *true* values, which are unknown in real-world systems. Therefore, the actual measurements $\boldsymbol{l}$ differ from the true ones by some unknown true *corrections* $\tilde{\boldsymbol{v}}$:

$$\boldsymbol{l} + \tilde{\boldsymbol{v}} = \tilde{\boldsymbol{l}} \tag{2.4.2}$$

As the true values are unknown, it will generally be worked with estimated values instead. They are marked with a hat and directly replace the true values in the equations, which leads to

$$\widehat{\boldsymbol{l}} = \boldsymbol{l} + \widehat{\boldsymbol{v}} = \boldsymbol{f}(\widehat{\boldsymbol{x}}). \tag{2.4.3}$$

The goal of parameter estimation is to find those parameters which explain the measurements best. Or, said differently, which minimize the estimated corrections

$$\widehat{\boldsymbol{v}} = \boldsymbol{f}(\widehat{\boldsymbol{x}}) - \boldsymbol{l}. \tag{2.4.4}$$

A common approach is to determine $\widehat{\boldsymbol{x}}$ such as the squared corrections become minimal. The estimation problem is then solved by

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ \widehat{\boldsymbol{v}}^T\widehat{\boldsymbol{v}} \tag{2.4.5}$$

where $\widehat{\boldsymbol{x}}$ is called the *ordinary least squares* solution.

2.4.1. Linear least squares

For a linear system

$$\boldsymbol{l} = \boldsymbol{f}(\boldsymbol{x}) = \mathrm{A}\boldsymbol{x} \tag{2.4.6}$$

where A is a $(n \times m)$ matrix. With $n > m$ the solution to (2.4.5) is determined as

$$\widehat{\boldsymbol{x}} = (\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\boldsymbol{l}. \tag{2.4.7}$$

To adjust the influence of single measurements it is possible to introduce a weight matrix W, such as the estimation problem from (2.4.5) changes to

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\ \widehat{\boldsymbol{v}}^T\mathrm{W}\widehat{\boldsymbol{v}}. \tag{2.4.8}$$

The solution is now

$$\widehat{\boldsymbol{x}} = (\mathrm{A}^T\mathrm{W}\mathrm{A})^{-1}\mathrm{A}^T\mathrm{W}\boldsymbol{l}. \tag{2.4.9}$$

A common choice for W is the inverse of the covariance matrix of the measurements:

$$\mathrm{W} = \Sigma_{\boldsymbol{ll}}^{-1}. \tag{2.4.10}$$

In this work, measurements are considered to be normally distributed. As a consequence, the minimization term is the squared *Mahalanobis distance*

$$\Omega^2 = \widehat{\boldsymbol{v}}^T\Sigma_{\boldsymbol{ll}}^{-1}\widehat{\boldsymbol{v}} \tag{2.4.11}$$

and the corresponding solution (2.4.9) is the *maximum likelihood estimate* (MLE). Remark: If the observations are mutually independent, the estimation process is in accordance with the *Gauss-Markov model*. In that case the covariance matrix only contains the variances $\sigma^2_{l_i l_i}$ on its diagonal, all other elements are 0. The weight matrix can then easily be determined by calculating the inverses $\frac{1}{\sigma^2_{l_i l_i}}$ of all non-zero elements.

2.4.2. Uncertainties of the linear least squares solution

To determine the uncertainty of the linear least squares solution, linear uncertainty propagation can be used. From (2.2.12) follows that for a linear system

$$\widehat{\boldsymbol{x}} = \mathrm{H}\boldsymbol{l} \tag{2.4.12}$$

the uncertainty $\Sigma_{\widehat{\boldsymbol{x}}\widehat{\boldsymbol{x}}}$ of $\widehat{\boldsymbol{x}}$ is determined via

$$\Sigma_{\widehat{\boldsymbol{x}}\widehat{\boldsymbol{x}}} = \mathrm{H}\Sigma_{\boldsymbol{ll}}\mathrm{H}^T. \tag{2.4.13}$$

Substituting $\mathrm{H} = (\mathrm{A}^T\mathrm{W}\mathrm{A})^{-1}\mathrm{A}^T\mathrm{W}$ and $\mathrm{W} = \Sigma_{\boldsymbol{ll}}^{-1}$ leads to

$$\Sigma_{\widehat{\boldsymbol{x}}\widehat{\boldsymbol{x}}} = \left[(\mathrm{A}^T\Sigma_{\boldsymbol{ll}}^{-1}\mathrm{A})^{-1}\mathrm{A}^T\Sigma_{\boldsymbol{ll}}^{-1}\right]\Sigma_{\boldsymbol{ll}}\left[(\mathrm{A}^T\Sigma_{\boldsymbol{ll}}^{-1}\mathrm{A})^{-1}\mathrm{A}^T\Sigma_{\boldsymbol{ll}}^{-1}\right]^T. \tag{2.4.14}$$

Many of the elements cancel out when applying the inverse and transpose operators. This finally delivers

$$\Sigma_{\widehat{\boldsymbol{x}}\widehat{\boldsymbol{x}}} = (\mathrm{A}^T\Sigma_{\boldsymbol{ll}}^{-1}\mathrm{A})^{-1} \tag{2.4.15}$$

for the covariance matrix of the estimated parameters.

2.4.3. Non-linear least squares

In case the model function $\boldsymbol{f}(\boldsymbol{x})$ is not linear in the parameters, the solution (2.4.9) can not be determined directly. Instead, an initial estimation $\widehat{\boldsymbol{x}}^a$ of the parameters is needed, at which $\boldsymbol{f}(\widehat{\boldsymbol{x}}) = \boldsymbol{l} + \widehat{\boldsymbol{v}}$ can be approximated locally by a linear function:

$$\Delta\boldsymbol{l} + \widehat{\boldsymbol{v}} = \mathrm{A}\widehat{\Delta\boldsymbol{x}} \tag{2.4.16}$$

with

$$\mathrm{A} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\widehat{\boldsymbol{x}}^a} \tag{2.4.17}$$

$$\Delta\boldsymbol{l} = \boldsymbol{l} - \boldsymbol{f}(\widehat{\boldsymbol{x}}^a) \tag{2.4.18}$$

$$\widehat{\Delta\boldsymbol{x}} = \widehat{\boldsymbol{x}} - \widehat{\boldsymbol{x}}^a. \tag{2.4.19}$$

The optimization problem (2.4.8) is then solved iteratively by first determining the parameter update

$$\widehat{\Delta\boldsymbol{x}} = (\mathrm{A}^T\Sigma_{ll}^{-1}\mathrm{A})^{-1}\mathrm{A}^T\Sigma_{ll}^{-1}\Delta\boldsymbol{l} \tag{2.4.20}$$

which is subsequently used to refine the approximated parameters:

$$\widehat{\boldsymbol{x}}^a_{\text{updated}} = \widehat{\boldsymbol{x}}^a + \widehat{\Delta\boldsymbol{x}}. \tag{2.4.21}$$

This is repeated until convergence criteria are met, e.g. until $|\widehat{\Delta \boldsymbol{x}}|$ drops below a specified threshold.
So far the measurements $\boldsymbol{l}$ were always explicitly given by the model function. In projective geometry, however, models often are implicit functions of both the observations and the parameters $\boldsymbol{x}$:

$$\boldsymbol{g}(\boldsymbol{l}, \boldsymbol{x}) = \mathbf{0} \quad \Leftrightarrow \quad \boldsymbol{g}(\widehat{\boldsymbol{l}}, \widehat{\boldsymbol{x}}) = \mathbf{0}. \tag{2.4.22}$$

Again initial estimates are needed to achieve linearization and to proceed with an iterative optimization procedure. In this case, also the measurements will be updated. This is expressed by using approximate measurements $\widehat{\boldsymbol{l}}^a$, where the actual measurements $\boldsymbol{l}$ serve as starting values. They are related to the estimated corrections via

$$\widehat{\boldsymbol{l}} = \boldsymbol{l} + \widehat{\boldsymbol{v}} = \widehat{\boldsymbol{l}}^a + \widehat{\Delta \boldsymbol{l}}. \tag{2.4.23}$$

This leads to the optimization problem

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \, (\widehat{\boldsymbol{l}}^a + \widehat{\Delta \boldsymbol{l}} - \boldsymbol{l})^T \mathrm{W} (\widehat{\boldsymbol{l}}^a + \widehat{\Delta \boldsymbol{l}} - \boldsymbol{l}). \tag{2.4.24}$$

It is solved with the linearized model function as a constraint:

$$\boldsymbol{g}(\widehat{\boldsymbol{l}}, \widehat{\boldsymbol{x}}) \approx \boldsymbol{g}(\widehat{\boldsymbol{l}}^a, \widehat{\boldsymbol{x}}^a) + \mathrm{A} \widehat{\Delta \boldsymbol{x}} + \mathrm{B}^T \widehat{\Delta \boldsymbol{l}} = \mathbf{0} \tag{2.4.25}$$

where

$$\mathrm{A} = \left. \frac{\partial \boldsymbol{g}(\boldsymbol{l}, \boldsymbol{x})}{\partial \boldsymbol{x}} \right|_{\substack{\boldsymbol{l} = \widehat{\boldsymbol{l}}^a \\ \boldsymbol{x} = \widehat{\boldsymbol{x}}^a}} \quad \mathrm{B}^T = \left. \frac{\partial \boldsymbol{g}(\boldsymbol{l}, \boldsymbol{x})}{\partial \boldsymbol{l}} \right|_{\substack{\boldsymbol{l} = \widehat{\boldsymbol{l}}^a \\ \boldsymbol{x} = \widehat{\boldsymbol{x}}^a}}. \tag{2.4.26}$$

Again, the covariance matrix of the measurements $\Sigma_{\boldsymbol{ll}}$ is used to minimize the weighted squared errors as in (2.4.8). This leads to the equation

$$\mathrm{A}^T (\mathrm{B}^T \Sigma_{\boldsymbol{ll}} \mathrm{B})^{-1} \mathrm{A} \widehat{\Delta \boldsymbol{x}} = \mathrm{A}^T (\mathrm{B}^T \Sigma_{\boldsymbol{ll}} \mathrm{B})^{-1} \boldsymbol{c}_g \tag{2.4.27}$$

with

$$\boldsymbol{c}_g = -\boldsymbol{g}(\widehat{\boldsymbol{l}}^a, \widehat{\boldsymbol{x}}^a) + \mathrm{B}^T (\widehat{\boldsymbol{l}}^a - \boldsymbol{l}). \tag{2.4.28}$$

From this, the least squares solution

$$\widehat{\Delta \boldsymbol{x}} = (\mathrm{A}^T (\mathrm{B}^T \Sigma_{\boldsymbol{ll}} \mathrm{B})^{-1} \mathrm{A})^{-1} \mathrm{A}^T (\mathrm{B}^T \Sigma_{\boldsymbol{ll}} \mathrm{B})^{-1} \boldsymbol{c}_g \tag{2.4.29}$$

$$\widehat{\Delta \boldsymbol{l}} = \Sigma_{\boldsymbol{ll}} \mathrm{B} (\mathrm{B}^T \Sigma_{\boldsymbol{ll}} \mathrm{B})^{-1} (\boldsymbol{c}_g - \mathrm{A} \widehat{\Delta \boldsymbol{x}}) - (\widehat{\boldsymbol{l}}^a - \boldsymbol{l}) \tag{2.4.30}$$

can be determined. Now parameters and measurements are updated via

$$\widehat{\boldsymbol{x}}^a_{\text{updated}} = \widehat{\boldsymbol{x}}^a + \widehat{\Delta \boldsymbol{x}} \tag{2.4.31}$$

$$\widehat{\boldsymbol{l}}^a_{\text{updated}} = \widehat{\boldsymbol{l}}^a + \widehat{\Delta \boldsymbol{l}} \tag{2.4.32}$$

and can be used to initialize the next iteration step. Again, this procedure is repeated until convergence. If the measurement noise is indeed normally distributed, still the Mahanalobis distance is minimized, which makes the final solution for $\widehat{\boldsymbol{x}}^a$ the maximum likelihood estimate. The uncertainties of the final parameters are described by the covariance matrix

$$\Sigma_{\widehat{\boldsymbol{x}}^a \widehat{\boldsymbol{x}}^a} = (\mathrm{A}^T (\mathrm{B}^T \Sigma_{\boldsymbol{ll}} \mathrm{B})^{-1} \mathrm{A})^{-1}. \tag{2.4.33}$$

2.4.4. MLE with reduced homogeneous coordinates

The last section showed how least squares solutions for implicit models that are non-linear in the parameters can be determined. Implicit constraints many times arise when working with homogeneous entities. A prerequisite for the described procedure to work is that the covariance matrix $\Sigma_{\boldsymbol{ll}}$ is of full rank. When using homogeneous representations of geometric entities, however, this is not the case. Take the following example as a demonstration: the uncertainty of the euclidean vector $\boldsymbol{x} = (x_1, x_2)^T$ is defined by a (2×2) covariance matrix $\Sigma_{\boldsymbol{xx}}$. Making the vector homogeneous by adding a 1 leads to $\mathbf{x} = (x_1, x_2, 1)^T$ and the (3×3) covariance matrix

$$\Sigma_{\mathbf{xx}} = \begin{bmatrix} \Sigma_{\boldsymbol{xx}} & \mathbf{0}_2 \\ \mathbf{0}_2^T & 0 \end{bmatrix}. \tag{2.4.34}$$

This is a singular matrix and therefore its inverse cannot be used as a weight matrix. Furthermore, the increased number of parameters of redundant representations requires the use of additional constraints. These lead to an increased number of equations and add more unknowns in form of Lagrangian multipliers. As additional constraints are many times non-linear, the bias created by linearization is also augmented.
To remedy these drawbacks, Förstner proposed in [Foe10] to use minimal representations of homogeneous entities for parameter estimation and also for the update procedure that arises from non-linear models. He calls these minimal representations *reduced coordinates* and chooses a tangent plane of the unit sphere in the corresponding space as the subspace they live in. For a 2D point $\boldsymbol{x}$, which is represented by a 3D homogeneous vector $\mathbf{x}$, the 2-sphere is used (see Figure 2.4.1 for an illustration). The corresponding realization of $\mathbf{x}$ is its projection to this sphere, which is determined by spherical normalization:

$$\mathbf{x}^s = \frac{\mathbf{x}}{|\mathbf{x}|}. \tag{2.4.35}$$

Position $\mathbf{x}^s$ defines the origin of the reduced space, any two perpendicular tangent vectors serve to define the coordinate axes. They are conveniently identified as the columns of the null spaces of the transposed homogeneous vectors. Therefore, the (3×2) matrix

$$\mathrm{J}_x = \mathrm{null}(\mathbf{x}^{sT}) \tag{2.4.36}$$

transforms points from 2D reduced to 3D homogeneous space:

$$\mathbf{x}^s = \mathrm{J}_x \boldsymbol{x}_r. \tag{2.4.37}$$

Here, $\boldsymbol{x}_r$ is the position in reduced coordinates on the tangent plane. As $\mathbf{x}^s$ has unit length, J_x is orthonormal and therefore (2.4.37) can easily be inverted to project homogeneous coordinates to the reduced space:

$$\boldsymbol{x}_r = \mathrm{J}_x^T \mathbf{x}^s. \tag{2.4.38}$$

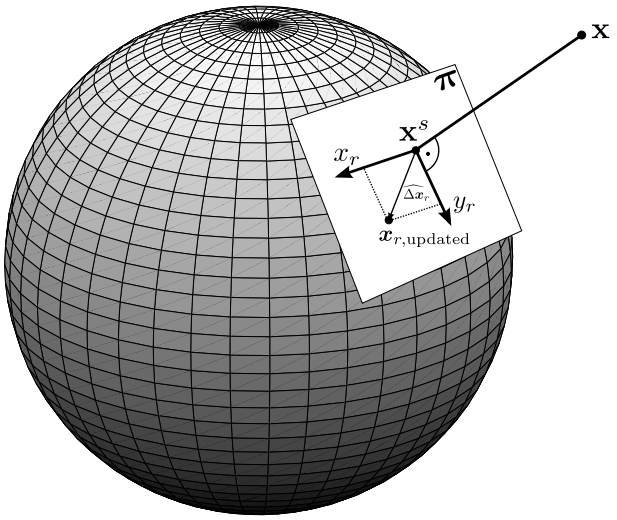


Figure 2.4.1.: Reduction of a homogeneous 2D point $\mathbf{x}$: two tangent vectors of the unit sphere at the piercing point $\mathbf{x}^s$ define the plane $\boldsymbol{\pi}$ which serves as reduced parameter space. Parameter updates $\widehat{\Delta \boldsymbol{x}_r}$ gained from MLE are determined here and used to calculate updated reduced parameters $\boldsymbol{x}_{r,\text{updated}}$.

As the actual intention is to perform maximum likelihood estimation in the reduced space, the covariance matrices also need to be transformed. In a first step, the uncertainties after spherical normalization are determined via

$$\Sigma_{\mathbf{x}^s\mathbf{x}^s} = \mathrm{J}_s \Sigma_{\mathbf{x}\mathbf{x}} \mathrm{J}_s^T \quad \text{with} \quad \mathrm{J}_s = \frac{1}{|\mathbf{x}|}(\mathrm{I}_3 - \mathbf{x}^s \mathbf{x}^{sT}). \tag{2.4.39}$$

The uncertainties in the reduced space are subsequently calculated as

$$\Sigma_{\boldsymbol{x}_r\boldsymbol{x}_r} = \mathrm{J}_x^T \Sigma_{\mathbf{x}^s\mathbf{x}^s} \mathrm{J}_x. \tag{2.4.40}$$

When solving an optimization problem like (2.4.24) in reduced coordinates it changes to

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \, (\widehat{\boldsymbol{l}}_r^a + \widehat{\Delta \boldsymbol{l}}_r - \boldsymbol{l}_r)^T \mathrm{W} (\widehat{\boldsymbol{l}}_r^a + \widehat{\Delta \boldsymbol{l}}_r - \boldsymbol{l}_r). \tag{2.4.41}$$

The constraints are now expressed as

$$\boldsymbol{g}(\widehat{\boldsymbol{l}}_r, \widehat{\boldsymbol{x}}_r) = \mathbf{0}. \tag{2.4.42}$$

Mostly, the constraints will be non-linear in parameters and measurements, which requires the determination of approximate values to start the iterative optimization procedure. Reduced approximate parameters $\boldsymbol{x}_r^a$ and their covariance matrix $\Sigma_{\boldsymbol{x}_r^a\boldsymbol{x}_r^a}$ are then determined as shown by (2.4.38) and (2.4.40), using the spherically normalized approximate coordinates $\mathbf{x}^{as}$ instead of $\mathbf{x}^s$. The representations for the measurements are calculated analogously from their spherically normalized versions $\mathbf{l}^{as}$ via

$$\boldsymbol{l}_r^a = \mathrm{J}_l^T \mathbf{l}^{as} \tag{2.4.43}$$

where the columns of J_l are the normalized coordinate axes of the reduced space in the original measurement space. Covariance matrices are then identified as

$$\Sigma_{\boldsymbol{l}_r^a \boldsymbol{l}_r^a} = \mathrm{J}_l^T \Sigma_{\mathbf{l}^s \mathbf{l}^s} \mathrm{J}_l. \tag{2.4.44}$$

For ease of notation, the superscript s is dropped from now on and all homogeneous entities are assumed to be spherically normalized.
To map the linearized constraint

$$\boldsymbol{g}(\widehat{\boldsymbol{l}}^a, \widehat{\boldsymbol{x}}^a) + \mathrm{A}\widehat{\Delta \boldsymbol{x}} + \mathrm{B}^T \widehat{\Delta \boldsymbol{l}} = \mathbf{0} \tag{2.4.45}$$

to the reduced coordinate system, the parameter and measurement updates need to be transformed. This is done with the help of the projection matrices J_x and J_l:

$$\widehat{\Delta \boldsymbol{x}} = \mathrm{J}_x \widehat{\Delta \boldsymbol{x}_r} \tag{2.4.46}$$

$$\Rightarrow \mathrm{A}\widehat{\Delta \boldsymbol{x}} = \mathrm{A}\mathrm{J}_x \widehat{\Delta \boldsymbol{x}_r} = \mathrm{A}_r \widehat{\Delta \boldsymbol{x}_r} \quad \text{with} \quad \mathrm{A}_r = \mathrm{A}\mathrm{J}_x, \tag{2.4.47}$$

$$\widehat{\Delta \boldsymbol{l}} = \mathrm{J}_l \widehat{\Delta \boldsymbol{l}_r} \tag{2.4.48}$$

$$\Rightarrow \mathrm{B}^T \widehat{\Delta \boldsymbol{l}} = \mathrm{B}^T \mathrm{J}_l \widehat{\Delta \boldsymbol{l}_r} = \mathrm{B}_r^T \widehat{\Delta \boldsymbol{l}_r} \quad \text{with} \quad \mathrm{B}_r^T = \mathrm{B}^T \mathrm{J}_l. \tag{2.4.49}$$

The resulting constraint in reduced coordinates reads as

$$\boldsymbol{g}(\widehat{\boldsymbol{l}}_r^a, \widehat{\boldsymbol{x}}_r^a) + \mathrm{A}_r \widehat{\Delta \boldsymbol{x}_r} + \mathrm{B}_r^T \widehat{\Delta \boldsymbol{l}_r} = \mathbf{0}. \tag{2.4.50}$$

Now the reduced updates are determined via

$$\widehat{\Delta \boldsymbol{x}_r} = (\mathrm{A}_r^T (\mathrm{B}_r^T \Sigma_{\boldsymbol{l}_r^a \boldsymbol{l}_r^a} \mathrm{B}_r)^{-1} \mathrm{A}_r)^{-1} \mathrm{A}_r^T (\mathrm{B}_r^T \Sigma_{\boldsymbol{l}_r^a \boldsymbol{l}_r^a} \mathrm{B}_r)^{-1} \boldsymbol{c}_{gr} \tag{2.4.51}$$

$$\widehat{\Delta \boldsymbol{l}_r} = \Sigma_{\boldsymbol{l}_r^a \boldsymbol{l}_r^a} \mathrm{B}_r (\mathrm{B}_r^T \Sigma_{\boldsymbol{l}_r^a \boldsymbol{l}_r^a} \mathrm{B}_r)^{-1} (\boldsymbol{c}_{gr} - \mathrm{A}_r \widehat{\Delta \boldsymbol{x}_r}) - (\widehat{\boldsymbol{l}}_r^a - \boldsymbol{l}_r) \tag{2.4.52}$$

with

$$\boldsymbol{c}_{gr} = -\boldsymbol{g}(\widehat{\boldsymbol{l}}_r^a, \widehat{\boldsymbol{x}}_r^a) + \mathrm{B}_r^T (\widehat{\boldsymbol{l}}_r^a - \boldsymbol{l}_r). \tag{2.4.53}$$

The actual update processes are then executed in the tangent plane via

$$\widehat{\mathbf{x}}_{updated}^a = \widehat{\mathbf{x}}^a + \mathrm{J}_x \widehat{\Delta \boldsymbol{x}_r} \tag{2.4.54}$$

$$\widehat{\mathbf{l}}_{updated}^a = \widehat{\mathbf{l}}^a + \mathrm{J}_l \widehat{\Delta \boldsymbol{l}_r}. \tag{2.4.55}$$

As $\widehat{\mathbf{x}}_{updated}^a$ and $\widehat{\mathbf{l}}_{updated}^a$ do no longer lie on the unit sphere, spherical normalization is needed to get the final new approximations $\widehat{\mathbf{x}}^a$ and $\widehat{\mathbf{l}}^a$ which are subsequently used to determine new tangent spaces.
In case of the measurements, another aspect has to be taken into account. The uncertainties reflected by $\Sigma_{\mathbf{l}^a \mathbf{l}^a}$ are those given by the user for the original observations, serving as initial estimates for the true measurements. As now a new estimate is determined, the

uncertainties also have to be adjusted. The transformation from the original measurements $\mathbf{l}^a$ to the current estimates $\widehat{\mathbf{l}}^a$ can be considered as a rotation within their common plane:

$$\widehat{\mathbf{l}}^a = \mathrm{R}_{\mathbf{l}^a\widehat{\mathbf{l}}^a}\mathbf{l}^a \tag{2.4.56}$$

(confer Section 2.1.4.3 for details on $\mathrm{R}_{\mathbf{l}^a\widehat{\mathbf{l}}^a}$).
Therefore, the covariance matrix $\Sigma_{\mathbf{l}^a\mathbf{l}^a}$ can no longer be constant along the optimization process, but has to change for each iteration. Consequently, $\Sigma_{\mathbf{l}_r^a\mathbf{l}_r^a}$ in (2.4.51) and (2.4.52) needs to be replaced by

$$\widehat{\Sigma}_{\mathbf{l}_r^a\mathbf{l}_r^a} = \mathrm{J}_{\mathbf{l}}^T\widehat{\Sigma}_{\mathbf{l}^a\mathbf{l}^a}\mathrm{J}_{\mathbf{l}} \tag{2.4.57}$$

with

$$\widehat{\Sigma}_{\mathbf{l}^a\mathbf{l}^a} = \mathrm{R}_{\mathbf{l}^a\widehat{\mathbf{l}}^a}\Sigma_{\mathbf{l}^a\mathbf{l}^a}\mathrm{R}_{\mathbf{l}^a\widehat{\mathbf{l}}^a}^T. \tag{2.4.58}$$

The update procedure is repeated until convergence. Uncertainties of the final reduced parameter updates $\widehat{\Delta\boldsymbol{x}_r}$ are described by the covariance matrix

$$\Sigma_{\widehat{\Delta\boldsymbol{x}_r}\widehat{\Delta\boldsymbol{x}_r}} = (\mathrm{A}_r^T(\mathrm{B}_r^T\widehat{\Sigma}_{\boldsymbol{l}_r^a\boldsymbol{l}_r^a}\mathrm{B}_r)^{-1}\mathrm{A}_r)^{-1}. \tag{2.4.59}$$

Transformation to the regular space via J_x delivers the covariance matrix

$$\Sigma_{\hat{\mathbf{x}}^a\hat{\mathbf{x}}^a} = \mathrm{J}_x\Sigma_{\widehat{\Delta\boldsymbol{x}_r}\widehat{\Delta\boldsymbol{x}_r}}\mathrm{J}_x^T \tag{2.4.60}$$

of the final parameter vector estimate $\widehat{\mathbf{x}}^a$.

2.4.4.1. MLE Example 1: best line intersection

In this section, the procedure described above will be applied to solve a simple estimation problem. This serves as an example which shows how maximum likelihood estimation with reduced homogeneous coordinates is used in practice.
The goal is to determine the best intersection point of multiple lines, i.e. the point which has the smallest distance to all lines. Incidence of a homogeneous 3D point $\mathbf{P} = [p_x\ p_y\ p_z\ p_h]^T$ and a line with Plücker coordinates $\mathbf{L}$ is given when $\overline{\Gamma}\mathbf{P} = \mathbf{0}$ ($\overline{\Gamma}$ is the dual Plücker matrix, confer Section 2.1.3 for the definition). This leads to the following constraint for each line

$$\boldsymbol{g}_i(\boldsymbol{l}_i, \boldsymbol{x}) = \boldsymbol{g}_i(\mathbf{L}_i, \mathbf{P}) = \overline{\Gamma}_i\mathbf{P} = \mathbf{0}. \tag{2.4.61}$$

The point $\mathbf{P}$ is the parameter vector which is to be determined and the spherically normalized line coordinates $\mathbf{L}_i^s$ with covariance matrices $\Sigma_{\mathbf{L}_i^s\mathbf{L}_i^s}$ are the measurements. The rank of the matrices $\overline{\Gamma}_i$ is 2, therefore two independent rows need to be chosen for each line, e.g. by selecting those containing the biggest absolute values. This leads to the smaller (2×4) matrices $\overline{\Gamma}_i^*$ which are stacked to form the matrix

$$\mathrm{A}^0 = \begin{pmatrix} \overline{\Gamma}_1^* \\ \overline{\Gamma}_2^* \\ \vdots \end{pmatrix}. \tag{2.4.62}$$

Now the initial approximation $\widehat{\mathbf{P}}^a$ of $\mathbf{P}$ can be identified as the singular vector belonging to the smallest singular value of A^0. The oberservations $\mathbf{L}_i^s$ serve as the initial measurement approximations $\widehat{\mathbf{L}}_i^a$, their covariance matrices are kept for now as $\Sigma_{\mathbf{L}_i^a\mathbf{L}_i^a} = \Sigma_{\mathbf{L}_i^s\mathbf{L}_i^s}$.
This allows to create the other elements needed for the optimization procedure:

$$\mathrm{A}_i = \left.\frac{\partial \boldsymbol{g}_i}{\partial \mathbf{P}}\right|_{\mathbf{L}_i=\widehat{\mathbf{L}}_i^a} = \overline{\Gamma}_i^a \tag{2.4.63}$$

$$\mathrm{B}_i^T = \left.\frac{\partial \boldsymbol{g}_i}{\partial \mathbf{L}_i}\right|_{\mathbf{P}=\widehat{\mathbf{P}}^a} = \begin{bmatrix} 0 & -p_z^a & p_y^a & -p_h^a & 0 & 0 \\ p_z^a & 0 & -p_x^a & 0 & -p_h^a & 0 \\ -p_y^a & p_x^a & 0 & 0 & 0 & -p_h^a \\ 0 & 0 & 0 & p_x^a & p_y^a & p_z^a \end{bmatrix} \tag{2.4.64}$$

$$\mathrm{J}_x = \mathrm{null}(\widehat{\mathbf{P}}^{aT}) \tag{2.4.65}$$

$$\mathrm{J}_{l,i} = \mathrm{null}([\widehat{\mathbf{L}}_i^a \ \widehat{\overline{\mathbf{L}}}_i^a]^T) \tag{2.4.66}$$

where $\overline{\mathbf{L}}$ are the dual Plücker coordinates of line coordinates $\mathbf{L}$ with direction and moment exchanged. J_x is a (4×3) matrix and the $\mathrm{J}_{l,i}$ are (6×4) matrices. After selecting appropriate rows of A_i and B_i^T to get A_i^* and B_i^{*T} (as before based on the entries of the $\overline{\Gamma}_i^a$), the reduced matrices can be determined via $\mathrm{A}_{i,r} = \mathrm{A}_i^*\mathrm{J}_x$, $\mathrm{B}_{i,r}^T = \mathrm{B}_i^{*T}\mathrm{J}_{l,i}$ and $\Sigma_{\mathbf{L}_{i,r}^a\mathbf{L}_{i,r}^a} = \mathrm{J}_{l,i}^T\Sigma_{\mathbf{L}_i^a\mathbf{L}_i^a}\mathrm{J}_{l,i}$. They are combined to form the final matrices

$$\mathrm{A}_r = \begin{bmatrix} \mathrm{A}_{1,r} \\ \mathrm{A}_{2,r} \\ \vdots \end{bmatrix}, \quad \mathrm{B}_r^T = \begin{bmatrix} \mathrm{B}_{1,r}^T & & 0 \\ & \mathrm{B}_{2,r}^T & \\ 0 & & \ddots \end{bmatrix} \tag{2.4.67}$$

and

$$\Sigma_{\boldsymbol{l}_r^a\boldsymbol{l}_r^a} = \begin{bmatrix} \Sigma_{\mathbf{L}_{1,r}^a\mathbf{L}_{1,r}^a} & & 0 \\ & \Sigma_{\mathbf{L}_{2,r}^a\mathbf{L}_{2,r}^a} & \\ 0 & & \ddots \end{bmatrix}. \tag{2.4.68}$$

Now equations (2.4.51) and (2.4.52) can be utilized to determine the reduced parameter and measurement update vectors $\widehat{\Delta\boldsymbol{x}}_r$ and $\widehat{\Delta\boldsymbol{l}}_r$. The actual updates are then executed as shown in (2.4.54) and (2.4.55) to get

$$\widehat{\mathbf{P}}_{updated}^a = \widehat{\mathbf{x}}_{updated}^a \quad \text{and} \quad \begin{pmatrix} \widehat{\mathbf{L}}_{1,updated}^a \\ \widehat{\mathbf{L}}_{2,updated}^a \\ \vdots \end{pmatrix} = \widehat{\mathbf{l}}_{updated}^a. \tag{2.4.69}$$

Spherical normalization of $\widehat{\mathbf{P}}_{updated}^a$ and the $\widehat{\mathbf{L}}_{i,updated}^a$ delivers the new approximations.
As mentioned before, also the covariance matrices $\Sigma_{\mathbf{L}_i^a\mathbf{L}_i^a}$ of the measurements have to be adjusted. The necessary (6×6) rotation matrices $\mathrm{R}_{\mathbf{L}_i\widehat{\mathbf{L}}_i^a}$ are determined from the current approximations $\widehat{\mathbf{L}}_i^a$ and the original observations $\mathbf{L}_i$ as shown in Section 2.1.4.3. This

allows to calculate the estimated measurement covariance matrices $\widehat{\Sigma}_{\widehat{\mathbf{L}}_i^a\widehat{\mathbf{L}}_i^a}$ via (2.4.58). With all values being updated, the next iteration of the optimization procedure starts. The procedure is repeated until convergence is reached, e.g. when the length of the parameter update vector is below a specified value.

2.4.4.2. MLE Example 2: 3D line fitting

Another application of MLE with reduced homogeneous coordinates is fitting a line through a set of 3D points. The points are given as $\boldsymbol{P}_i = [p_x\ p_y\ p_z]^T$ with uncertainties described by their covariance matrices $\Sigma_{\boldsymbol{P}_i\boldsymbol{P}_i}$, $i = 1..n_i$. To increase numerical stability it is recommended to condition these measurements. An appropriate method is to shift them by their mean position $\overline{\boldsymbol{P}}$ such as their center of gravity lies at the origin and scale them to have a maximum length of 1. With $s_p = \max_i |\boldsymbol{P}_i - \overline{\boldsymbol{P}}|$, the conditioned measurements are

$$\boldsymbol{P}'_i = \frac{1}{s_p}(\boldsymbol{P}_i - \overline{\boldsymbol{P}}) \quad \text{with} \quad \Sigma_{\boldsymbol{P}'_i\boldsymbol{P}'_i} = \frac{1}{s_p^2}\Sigma_{\boldsymbol{P}_i\boldsymbol{P}_i}. \tag{2.4.70}$$

These are transferred to their homogeneous equivalents

$$\mathbf{P}_i = [\boldsymbol{P}_i'^T\ 1]^T \quad \text{and} \quad \Sigma_{\mathbf{P}_i\mathbf{P}_i} = \begin{bmatrix} \Sigma_{\boldsymbol{P}'_i\boldsymbol{P}'_i} & \mathbf{0}_3 \\ \mathbf{0}_3^T & 0 \end{bmatrix}. \tag{2.4.71}$$

Spherical normalization delivers the final measurements $\boldsymbol{l}_i$ used to start the estimation process:

$$\boldsymbol{l}_i = \mathbf{P}_i^s = \frac{\mathbf{P}_i}{|\mathbf{P}_i|} = [p_{1,i}\ p_{2,i}\ p_{3,i}\ p_{4,i}]^T \quad \text{and} \quad \Sigma_{\boldsymbol{l}_i\boldsymbol{l}_i} = \Sigma_{\mathbf{P}_i^s\mathbf{P}_i^s} = \mathrm{J}_{s,i}\Sigma_{\mathbf{P}_i\mathbf{P}_i}\mathrm{J}_{s,i}^T \tag{2.4.72}$$

with

$$\mathrm{J}_{s,i} = \frac{1}{|\mathbf{P}_i|}(\mathrm{I}_4 - \mathbf{P}_i^s\mathbf{P}_i^{sT}). \tag{2.4.73}$$

To improve readability, the subscript s will be dropped from now on.
The constraints for the line fitting procedure are defined implicitly by the incidence condition of points and lines as

$$\boldsymbol{g}_i(\boldsymbol{l}_i, \boldsymbol{x}) = \boldsymbol{g}_i(\mathbf{P}_i, \mathbf{L}) = \overline{\Gamma}\mathbf{P}_i. \tag{2.4.74}$$

The parameters $\boldsymbol{x}$ are the elements $\{L_1, ..., L_6\}$ of $\mathbf{L}$, which can be found as the contents of the dual Plücker matrix

$$\overline{\Gamma} = \begin{bmatrix} 0 & L_3 & -L_2 & -L_4 \\ -L_3 & 0 & L_1 & -L_5 \\ L_2 & -L_1 & 0 & -L_6 \\ L_4 & L_5 & L_6 & 0 \end{bmatrix}. \tag{2.4.75}$$

Defining the approximate observation estimations $\widehat{\mathbf{P}}_i^a = \mathbf{P}_i$ and their covariance matrices $\Sigma_{\mathbf{P}_i^a\mathbf{P}_i^a}$ and assuming that an initial approximate estimate $\widehat{\mathbf{L}}^a$ is given, leads to the Jacobians

$$\mathrm{A}_i = \left.\frac{\partial \boldsymbol{g}_i}{\partial \mathbf{L}}\right|_{\mathbf{P}_i=\widehat{\mathbf{P}}_i^a} = \begin{bmatrix} 0 & -p_{3,i}^a & p_{2,i}^a & -p_{4,i}^a & 0 & 0 \\ p_{3,i}^a & 0 & -p_{1,i}^a & 0 & -p_{4,i}^a & 0 \\ -p_{2,i}^a & p_{1,i}^a & 0 & 0 & 0 & -p_{4,i}^a \\ 0 & 0 & 0 & p_{1,i}^a & p_{2,i}^a & p_{3,i}^a \end{bmatrix} \tag{2.4.76}$$

$$\mathrm{B}_i^T = \left.\frac{\partial \boldsymbol{g}_i}{\partial \mathbf{P}_i}\right|_{\mathbf{L}=\widehat{\mathbf{L}}^a} = \overline{\Gamma}^a. \tag{2.4.77}$$

Because the rank of $\overline{\Gamma}$ is two, half of the rows need to be discarded to get linearly independent constraints. This can be done by selecting those two rows which have the largest entries to form the row-reduced matrices B_i^{*T} and A_i^*.
To obtain the reduced model, i.e. to map the optimization procedure to the tangent spaces of the corresponding parameters and measurements, the transformation matrices

$$\mathrm{J}_x = \mathrm{null}\left([\widehat{\mathbf{L}}^a\ \widehat{\overline{\mathbf{L}}}^a]^T\right) \quad \text{and} \quad \mathrm{J}_{l,i} = \mathrm{null}\left(\widehat{\mathbf{P}}_i^a\right) \tag{2.4.78}$$

are used:

$$\mathrm{A}_{i,r} = \mathrm{A}_i^* \mathrm{J}_x \tag{2.4.79}$$

$$\mathrm{B}_{i,r}^T = \mathrm{B}_i^{*T} \mathrm{J}_{l,i} \tag{2.4.80}$$

$$\Sigma_{\mathbf{P}_{i,r}^a\mathbf{P}_{i,r}^a} = \mathrm{J}_{l,i}^T \Sigma_{\mathbf{P}_i^a\mathbf{P}_i^a} \mathrm{J}_{l,i}. \tag{2.4.81}$$

This leads to the final matrices needed to determine the parameter and measurement updates in reduced space:

$$\mathrm{A}_r = [A_{1,r}\ A_{2,r}\ \ldots\ A_{n_i,r}]^T \tag{2.4.82}$$

$$\mathrm{B}_r^T = \begin{bmatrix} \mathrm{B}_{1,r}^T & & & 0 \\ & \mathrm{B}_{2,r}^T & & \\ & & \ddots & \\ 0 & & & \mathrm{B}_{n_i,r}^T \end{bmatrix} \tag{2.4.83}$$

$$\Sigma_{\boldsymbol{l}_r^a\boldsymbol{l}_r^a} = \begin{bmatrix} \Sigma_{\mathbf{P}_{1,r}^a\mathbf{P}_{1,r}^a} & & & 0 \\ & \Sigma_{\mathbf{P}_{2,r}^a\mathbf{P}_{2,r}^a} & & \\ & & \ddots & \\ 0 & & & \Sigma_{\mathbf{P}_{n_i,r}^a\mathbf{P}_{n_i,r}^a} \end{bmatrix}. \tag{2.4.84}$$

The reduced updates $\widehat{\Delta \boldsymbol{x}_r}$ and $\widehat{\Delta \boldsymbol{l}_r}$ are determined as shown in (2.4.51) and (2.4.52). Updating the parameters and measurements via (2.4.54) and (2.4.55) leads to

$$\widehat{\mathbf{L}}^{a'}_{updated} = \widehat{\mathbf{x}}^{a}_{updated} \quad \text{and} \quad \begin{pmatrix} \widehat{\mathbf{P}}^{a}_{1,updated} \\ \widehat{\mathbf{P}}^{a}_{2,updated} \\ \vdots \\ \widehat{\mathbf{P}}^{a}_{n_i,updated} \end{pmatrix} = \widehat{\mathbf{l}}^{a}_{updated}. \tag{2.4.85}$$

So far, $\widehat{\mathbf{L}}^{a'}_{updated}$ is a general 6D vector. It is therefore not guaranteed to be a valid line representation, as it does not necessarily fulfill the Plücker constraint (2.1.5). This can be remedied by the procedure described in Section 2.1.3 to obtain the Plücker coordinates $\widehat{\mathbf{L}}^{a}_{updated}$. Subsequent spherical normalization gives the new estimated approximations for the line parameters $\widehat{\mathbf{L}}^{a}$ and all line points $\widehat{\mathbf{P}}^{a}_{i}$. Before starting the next iteration, the covariance matrices of the observations $\Sigma_{\mathbf{P}^a_i \mathbf{P}^a_i}$ have to be updated as well. Section 2.1.4.3 shows how the (4×4) rotation matrices $\mathrm{R}_{\mathbf{P}_i \widehat{\mathbf{P}}^a_i}$ can be determined which are required to calculate the estimated measurement covariance matrices $\widehat{\Sigma}_{\widehat{\mathbf{P}}^a_i \widehat{\mathbf{P}}^a_{i,r}}$ by using (2.4.58). Now the next iteration of the optimization procedure can be started. When convergence is reached, the covariance matrix $\Sigma_{\widehat{\mathbf{L}}^a \widehat{\mathbf{L}}^a}$ of the line parameter estimate $\widehat{\mathbf{L}}^{a}$ can be obtained via (2.4.59) and (2.4.60).
In the beginning, the measurements were translated to lie around the origin to improve numerical stability. The corresponding mapping for Plücker line coordinates is executed by multiplying with the transformation matrix H_T (confer (2.1.12)). Inversion of this matrix allows to shift the line back to its original position and delivers the final (unconditioned) line estimate

$$\widehat{\mathbf{L}}^{a}_{f} = \mathrm{H}_T^{-1} \widehat{\mathbf{L}}^{a} \tag{2.4.86}$$

and (by linear uncertainty propagation) the corresponding final covariance matrix

$$\Sigma_{\widehat{\mathbf{L}}^a_f \widehat{\mathbf{L}}^a_f} = \mathrm{H}_T^{-1} \Sigma_{\widehat{\mathbf{L}}^a \widehat{\mathbf{L}}^a} \mathrm{H}_T^{-T}. \tag{2.4.87}$$

This concludes the chapter on theoretical basics. All further chapters will many times refer to the concepts explained here. The next chapter continuous with the introduction of different standard camera models and their calibration.

Chapter 3

Camera models

When using digital cameras for optical measurement purposes, a mathematical model must be found that describes the mapping between the 3D world where the measurements are taken and the image space that is the 2D projection of the scene. Depending on the application, the model has to provide at least one of the main functionalities: forward or back projection (compare Figure 1.0.1). Forward projection is the process of mapping a 3D world point to the image plane. It is commonly used for tasks like image rectification (i.e. the reversal of non-linear distortion effects) and bundle adjustment, which is applied during camera calibration or visual odometry determination. Back projection, on the other hand, delivers the locus of points, which is projected to a specified point in the image plane. This is usually, but not necessarily, a single line and will be referred to as *viewing ray*.
The imaging process can be as simple as a linear projection in the case of a perfect pinhole camera. In that case, all viewing rays pass through a single point. Such a system is called a system with a *single point of view* (sometimes: *single effective viewpoint*) or also *central system*. By adding further hardware components, the process becomes more and more involved. These components can be (a combination of) optical lenses, digital imaging sensors or mirrors of different shapes. An imaging system that combines a standard camera and a mirror is called a *catadioptric* system. Depending on its setup, formulating algebraic rules for forward and back projection can be challenging. Back projection can in general be easily solved by using the method of ray-tracing. Forward projection, on the other hand, is more involved, especially for *non-central* setups where no single point of view exists.
In the field of computer vision, the imaging system is commonly considered to be fixed, i.e. its configuration and therefore the parameters of the chosen mathematical model are constant over time. This is an assumption also made in this work. Furthermore, optical phenomena like lens aberrations or scattering (e.g. on mirror surfaces) are not considered. It will also be assumed that the depth of field is large enough to have all objects of rel-

evance in focus, i.e. there is no visible circle of confusion at any time. Depending on the actual configuration of the imaging system in terms of lens thickness, aperture, field of view, sensor resolution and/or intended purpose, these assumptions lead to appropriate approximations of real imaging systems. In [Han11] Hanning gives a concise overview of these aspects and their influence on camera models and calibration procedures.
As there are many different kinds of camera models, Sturm et al. propose dividing them into three categories: global, local and discrete camera models (see [Stu10]). While global models use a limited set of parameters that affect the camera mapping for the complete field of view, local models have parameters that describe the imaging process only in a subset of the field of view. Discrete models, on the other hand, parameterize the mapping process for each pixel separately. In this work, the concepts arising from discrete models will be used to create a local model. This offers great flexibility and therefore allows a great variety of imaging systems to be described, including non-central ones and those with locally changing distortions. At the same time, the basic functionalities of general back and forward projection are provided, improving the model's usability compared to that of a discrete model.

This chapter will introduce several camera models, such as the pinhole camera model in Section 3.1, omnidirectional models in Section 3.2 and generic models in Section 3.3. Each section also covers the calibration of the explained model. One of the goals is to convey the complexity of the tasks of camera modeling and calibration. Simplifying the process by providing a method to describe, calibrate and use the generic camera model proposed by Grossberg and Nayar (see Section 3.3.2) is the main contribution of this work. The corresponding ideas and concepts will be described in subsequent chapters.

3.1. The pinhole camera model

The pinhole camera model describes the imaging process of a camera obscura. Fig. 3.1.1 shows an illustration of the process and all relevant parameters. Light rays that are reflected by a scene at a point $\boldsymbol{P}_{cam}$ pass through an ideally infinitely small *pinhole*, which is also called *optical center* or *center of projection*. In a real camera, all rays that come from a scene intersect a plane which lies behind the pinhole, where they form an inverted image. The plane is called *image plane* and its distance to the pinhole is the *focal length* f. For mathematical reasons, i.e. to avoid inversion of the coordinates, the image plane is often virtually placed in front of the optical center in the pinhole model, also with a distance of f. The ray that is perpendicular to the image plane and passes through the optical center is the *principal axis* of the camera. Its intersection point with the image plane is the *principal point* $\boldsymbol{p}_p$. Two coordinate systems are important when describing

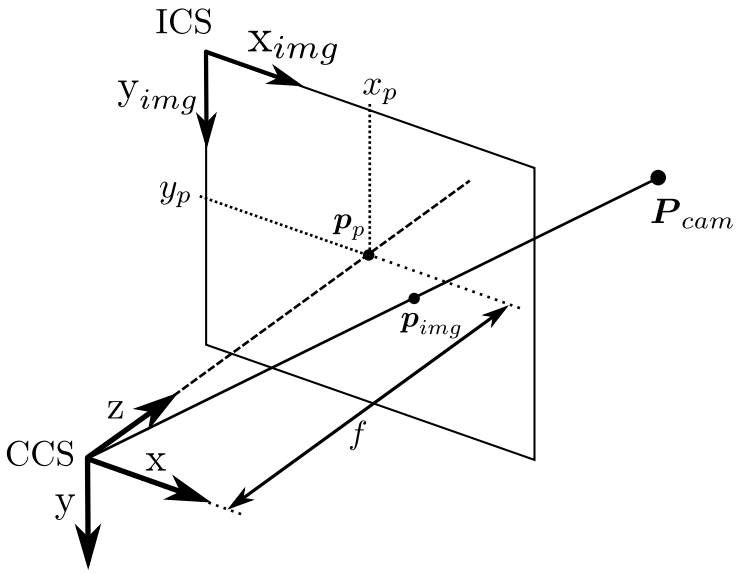


Figure 3.1.1.: The pinhole model allows a point $\boldsymbol{P}_{cam}$ to be projected from the camera coordinate system (CCS) to $\boldsymbol{p}_{img}$ in the image coordinate system (ICS). Also shown are the principle point $\boldsymbol{p}_p = (x_p, y_p)^T$ and the focal length f.

digital cameras: the 3D camera coordinate system (CCS) has the principal axis as its z axis, and the xy-plane is parallel to the image plane and has an origin that lies in the optical center. The 2D image coordinate system (ICS) is the coordinate system of the imaging sensor. In this system, coordinates are given in picture elements (short: pixels) instead of meters as in the CCS. Forward projection of a point $\boldsymbol{P}_{cam} = (x, y, z)^T$ from the CCS to the ICS is done by first determining the intersection point $\boldsymbol{P}_{img}$ of the corresponding viewing ray with the virtual image plane. This point can be calculated using the concept of similar triangles:

$$\boldsymbol{P}_{img} = \begin{pmatrix} f\frac{x}{z} \\ f\frac{y}{z} \\ f \end{pmatrix}. \tag{3.1.1}$$

The first two components stand for the position of the projected point in the local 2D coordinate system. Dividing this by the size s of a sensor element and shifting it such as the principal point lies at $(x_p, y_p)^T$ gives the final pixel coordinates $\boldsymbol{p}_{img}$ in the image:

$$\boldsymbol{p}_{img} = \begin{pmatrix} f\frac{x}{sz} + x_p \\ f\frac{y}{sz} + y_p \end{pmatrix}. \tag{3.1.2}$$

When switching to homogeneous coordinates, i.e. $\mathbf{P}_{cam} = (x, y, z, 1)^T$ is to be mapped to the image plane, the matrix equation

$$\mathbf{p}_{img} = \begin{pmatrix} f\frac{x}{s} + x_p z \\ f\frac{y}{s} + y_p z \\ z \end{pmatrix} \tag{3.1.3}$$

$$= \begin{bmatrix} \frac{f}{s} & 0 & x_p & 0 \\ 0 & \frac{f}{s} & y_p & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.1.4}$$

$$= [\mathrm{K} \quad \mathbf{0}_3]\, \mathbf{P}_{cam} \tag{3.1.5}$$

can be used. $\mathbf{0}_3$ is a 3D column vector of zeros and

$$\mathrm{K} = \begin{bmatrix} \frac{f}{s} & 0 & x_p \\ 0 & \frac{f}{s} & y_p \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & x_p \\ 0 & \alpha & y_p \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1.6}$$

is called the *camera matrix*, which contains all the internal or *intrinsic* parameters of a pinhole camera. Dividing $\mathbf{p}_{img}$ in (3.1.3) by the last element and subsequently keeping only the first two elements delivers the same 2D position as shown in (3.1.2).
It is important to note that sometimes K is defined as

$$\mathrm{K} = \begin{bmatrix} \frac{f}{s_u} & c & x_p \\ 0 & \frac{f}{s_v} & y_p \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1.7}$$

where s_u and s_v are the different sizes of a sensor element in x and y direction and c is a *skew* parameter that is necessary in the rare case that the imaging sensor elements are not rectangular. In this work, however, it will be assumed that $c = 0$ and $s_u = s_v = s$.
Oftentimes, 3D points are not given in the camera coordinate system, but in a different one. Therefore, these points need to be transformed to the CCS before they can be mapped to the image coordinates. When using homogeneous coordinates, this can also be done by matrix multiplication. The matrix

$$\mathrm{T} = \begin{bmatrix} \mathrm{R} & \boldsymbol{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{3.1.8}$$

is called a homogeneous transformation matrix. It comprises a (3×3) rotation matrix R and a translation vector $\boldsymbol{t}$. In this work, the notation ${}^b\mathrm{T}_a$ symbolizes the transformation of a point from coordinate system a to coordinate system b. Multiplying a point $\mathbf{P}_a$ from the right side transforms it from system a to system b, i.e. $\mathbf{P}_b = {}^b\mathrm{T}_a \mathbf{P}_a$.
Following this notation, a point $\mathbf{P}_v$ given in a coordinate system v can first be mapped to the camera coordinate system c and subsequently to the image coordinate system via

$$\mathbf{p}_{img} = [K\ \mathbf{0}_3]\, {}^c\mathrm{T}_v \mathbf{P}_v. \tag{3.1.9}$$

When used in this context, the components of ${}^{c}\mathrm{T}_{v}$ are called external or *extrinsic* camera parameters. Combining intrinsic camera matrix K and extrinsic parameters ${}^{c}\mathrm{T}_{v}$ delivers

$$\mathbf{p}_{img} = \mathrm{P}\,\mathbf{P}_{v} \tag{3.1.10}$$

with

$$\mathrm{P} = [K\ \mathbf{0}_3]{}^{c}\mathrm{T}_{v}. \tag{3.1.11}$$

Matrix P is sometimes referred to as the *projection matrix* of the general linear pinhole camera. This concludes the introduction of the pinhole camera and its parameters. The next section shows how these parameters can be determined for a real system, a process which is called *camera calibration*.

3.1.1. Linear pinhole camera calibration

In the previous section, all parameters of the pinhole camera model were introduced. This section explains how they can be determined for a real camera system. The procedure will be used later during the calibration process proposed in this work.
For pinhole cameras without non-linear lens distortions, the intrinsic parameters can be determined by solving a system of linear equations. The only necessary equipment is a calibration pattern that has detectable visible features with known positions in a local coordinate system. Placing these patterns in front of the camera and taking images of them provides all the information needed to execute the calibration procedure.
Single-shot methods commonly need a three-dimensional calibration pattern (e.g.[Tsa87]). However, many procedures use a planar pattern and take multiple images of it from different positions and angles. This method simplifies the calibration process for the user as the relative poses do not need to be known, which allows the patterns and camera to be placed by hand.
For ease of notation (3.1.11) and (3.1.8) are used without subscripts to rewrite (3.1.10) as

$$\begin{aligned}\mathbf{p} &= \mathrm{P}\,\mathbf{P} && (3.1.12)\\ &= \mathrm{K}\,[\mathrm{R} \quad \boldsymbol{t}]\mathbf{P} && (3.1.13)\\ &= \mathrm{K}\,[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad \boldsymbol{t}]\mathbf{P} && (3.1.14)\end{aligned}$$

where $\boldsymbol{r}_i$ is the ith column vector of R. $\mathbf{P}$ is now a point on a calibration object with known local position. When the calibration object is planar, the object coordinate system can be placed on the corresponding plane in 3D space, with its x and y axis parallel to that plane. In that case, the z component of the calibration point $\mathbf{P}$ becomes 0. This leads

to

$$\mathbf{p} = \mathrm{K}\,[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad \boldsymbol{t}] \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} \tag{3.1.15}$$

$$= \mathrm{K}\,[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}] \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.1.16}$$

$$= \mathrm{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{3.1.17}$$

In this case

$$\mathrm{H} = \mathrm{K}\,[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}]. \tag{3.1.18}$$

It is a (3×3) homography that maps points on a calibration plane to points in the image plane and can be determined by using the *direct linear transform* (DLT) algorithm described by Hartley and Zisserman in [HZ03]. Due to its homogeneous nature, the equality in (3.1.18) is only up to scale. This allows an arbitrary scaling factor λ to be inserted:

$$\mathrm{H} = \lambda \mathrm{K}\,[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}]. \tag{3.1.19}$$

The main goal of the calibration procedure is to calculate the intrinsic parameters, which are the components of K. By separating H into its columns $\boldsymbol{h}_1$, $\boldsymbol{h}_2$ and $\boldsymbol{h}_3$ and exploiting the fact that $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are orthonormal, two constraints on the elements of K can be derived:

$$[\boldsymbol{h}_1 \quad \boldsymbol{h}_2 \quad \boldsymbol{h}_3] = \lambda \mathrm{K}\,[\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}] \tag{3.1.20}$$

$$\Rightarrow \quad \boldsymbol{h}_1^T \mathrm{K}^{-T} \mathrm{K}^{-1} \boldsymbol{h}_2 = 0 \tag{3.1.21}$$

$$\wedge \quad \boldsymbol{h}_1^T \mathrm{K}^{-T} \mathrm{K}^{-1} \boldsymbol{h}_1 = \boldsymbol{h}_2^T \mathrm{K}^{-T} \mathrm{K}^{-1} \boldsymbol{h}_2. \tag{3.1.22}$$

This allows

$$\omega = \mathrm{K}^{-T}\mathrm{K}^{-1} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \tag{3.1.23}$$

to be defined. With K as derived in (3.1.6), all elements of ω except ω_{11}, ω_{22}, ω_{13}, ω_{23} and ω_{33} become zero and $\omega_{11} = \omega_{22}$. That leaves 4 unknown parameters. Now, (3.1.21) and (3.1.22) can be rewritten as

$$\begin{aligned}(h_{11}h_{12} + h_{21}h_{22})\omega_{11} + (h_{11}h_{32} + h_{31}h_{12})\omega_{13} \\ + (h_{31}h_{22} + h_{21}h_{32})\omega_{23} + h_{31}h_{32}\omega_{33} = 0\end{aligned} \tag{3.1.24}$$

$$\begin{aligned}(h_{11}^2 + h_{21}^2 - h_{12}^2 - h_{22}^2)\omega_{11} + 2(h_{11}h_{31} - h_{12}h_{32})\omega_{13} \\ + 2(h_{21}h_{31} - h_{22}h_{32})\omega_{23} + h_{31}^2 - h_{32}^2\omega_{33} = 0\end{aligned} \tag{3.1.25}$$

which gives two independent linear constraints on the unknown parameters. As can be seen in (3.1.20), the relation between H and K is only defined up to scale, which leaves three free parameters to be estimated. Therefore, at least one more constraint is needed. Additional constraints can be gained from further images of the planar calibration pattern in different poses. Zhang shows in [Zha02] the importance of changing the orientation of the plane. A simple translation is not sufficient as the resulting constraints would be linearly dependent on those of the other configuration. For a thorough elaboration on singular configurations and the amount of constraints needed when adding more parameters and/or assuming that some of them are known, please refer to [SM99]. With the necessary requirements fulfilled, multiple additional equations like (3.1.24) and (3.1.25) can be stacked in a matrix A_ω to form the equation system

$$\mathrm{A}_\omega \begin{pmatrix} \omega_{11} \\ \omega_{13} \\ \omega_{23} \\ \omega_{33} \end{pmatrix} = 0. \tag{3.1.26}$$

It can be solved exactly when A_ω has 4 rows, or in a least squares manner in case it has more. This specifies a minimum of two calibration images in order for the procedure to succeed. Having determined the free parameters of ω, the desired intrinsic parameters (compare (3.1.6)) are computed as

$$x_p = -\frac{\omega_{13}}{\omega_{11}} \tag{3.1.27}$$

$$y_p = -\frac{\omega_{23}}{\omega_{11}} \tag{3.1.28}$$

$$\alpha = \frac{\sqrt{\omega_{11}\omega_{33} - \omega_{13}^2 - \omega_{23}^2}}{\omega_{11}}. \tag{3.1.29}$$

Now, the intrinsic camera parameters are known. Oftentimes, the poses of the calibration planes are also needed. They can be directly determined from (3.1.20) as

$$\begin{aligned} \boldsymbol{r}_1 &= \lambda \mathrm{K}^{-1}\boldsymbol{h}_1 \\ \boldsymbol{r}_2 &= \lambda \mathrm{K}^{-1}\boldsymbol{h}_2 \\ \boldsymbol{r}_3 &= \boldsymbol{r}_1 \times \boldsymbol{r}_2 \\ \boldsymbol{t} &= \lambda \mathrm{K}^{-1}\boldsymbol{h}_3. \end{aligned} \tag{3.1.30}$$

Since the rotational vectors have to be of unit length, the scaling factor λ is calculated via

$$\lambda = \frac{1}{2}\left(\frac{1}{||\mathrm{K}^{-1}\boldsymbol{h}_1||} + \frac{1}{||\mathrm{K}^{-1}\boldsymbol{h}_2||}\right). \tag{3.1.31}$$

In general, the vectors of the rotation matrix will not be orthogonal after this estimation procedure. Rotating $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ in their common plane around $\boldsymbol{r}_3$ can remedy this.
This concludes the task of linear camera calibration. A subsequent step is usually the execution of a non-linear optimization procedure, e.g. to determine lens distortion parameters. This will be elaborated on in the following section.

3.1.2. The pinhole camera with lens distortions

The last section showed how a pinhole camera can be calibrated by using a linear calibration procedure. For most cameras, however, the light travels through various lenses before reaching the image plane, introducing non-linear distortions to the imaging process. These distortions are commonly modeled in the 2D image coordinate system. Forward projection can therefore be considered as a two-step process comprised of a linear projection of a 3D point to the image plane and a subsequent non-linear distortion procedure. Many distortion models have been proposed, most of which are based on the Brown-Conrady model ([Bro66],[Con19]). Generally, a rotationally invariant part is considered, which is only dependent on the distance of a pixel from the principal point. The widely used models of Tsai [Tsa87] and Zhang [Zha02] are good examples of that. A distorted position $\boldsymbol{p}_d$ of a projected point $\boldsymbol{p}_u$ is determined by

$$\boldsymbol{p}_d = \left(1 + \sum_{i=1}^{N} k_{r_i} ||\boldsymbol{p}_u||^{2i}\right) \boldsymbol{p}_u \tag{3.1.32}$$

where the coefficients k_{r_i} are the radial distortion parameters. $\boldsymbol{p}_u = (x_u, y_u)^T = (\frac{x}{z}, \frac{y}{z})^T$ signifies in this case not the pixel position $\boldsymbol{p}_i$ as in (3.1.2) but the intersection of the viewing ray through point $\boldsymbol{P}_c = (x, y, z)^T$ with a plane defined by $z = 1$. Therefore, $||\boldsymbol{p}_u||$ is the distance from the principal point, which is also the center of the radial distortion. Setting N to a value not bigger than 3 is usually sufficient for modeling the distortions of off-the-shelf cameras. By using only even exponents, the square root of the calculation of $||\boldsymbol{p}_u||$ is directly eliminated, which has computational benefits.
Sometimes a translational part is added to the distortion function as proposed by Heikkilä and Silvén in [HS97]. Then, the distorted position is determined via

$$\boldsymbol{p}_{d_2} = \boldsymbol{p}_d + \begin{pmatrix} 2k_{t_1} x_u y_u + k_{t_2}(||\boldsymbol{p}_u||^2 + 2x_u^2) \\ 2k_{t_2} x_u y_u + k_{t_1}(||\boldsymbol{p}_u||^2 + 2y_u^2) \end{pmatrix} \tag{3.1.33}$$

with k_{t_1} and k_{t_2} being tangential distortion parameters. The final image pixel position is obtained by creating a homogeneous vector from the distorted position and using the camera matrix K for scaling and shifting:

$$\mathbf{p}_{img} = \mathrm{K} \begin{pmatrix} x_{d_2} \\ y_{d_2} \\ 1 \end{pmatrix}. \tag{3.1.34}$$

The set of distortion coefficients $\boldsymbol{k} = \{k_{r_1}, ..., k_{r_N}, k_{t_1}, k_{t_1}\}$ now also belongs to the set of intrinsic parameters. In general, keeping the size of $\boldsymbol{k}$ to a minimum is advisable to increase the stability of the estimation process.
Determining the final values for all parameters is done by executing a non-linear optimization procedure. Usually, the so-called *reprojection error* ϵ_r is minimized, which is the

summed distance between image point measurements $\boldsymbol{p}_{vj}$ (e.g. chessboard corners) and the mapped positions of the corresponding 3D points $\boldsymbol{P}_{vj}$ to the image:

$$\epsilon_r = \sum_{v=1}^{V} \sum_{j=1}^{J_v} [\boldsymbol{p}_{vj} - m(\boldsymbol{k}, \mathrm{K}, {}^c\mathrm{T}_v, \boldsymbol{P}_{vj})]. \tag{3.1.35}$$

Here, $m(\boldsymbol{x})$ symbolizes the mapping from world points to the camera image, utilizing the specified set of parameters $\boldsymbol{x}$. V is the number of calibration images or *views* and J_v is the amount of image features seen in image v. The intrinsic parameters $\boldsymbol{k}$ and K are assumed to be constant for all images. ${}^c\mathrm{T}_v$ comprises the extrinsic parameters R_v and $\boldsymbol{t}_v$ of each specific view v.
Since m is non-linear in the model parameters, a non-linear optimization procedure needs to be executed to minimize ϵ_r. Therefore, initial estimates are needed for all parameters. These are usually obtained by ignoring the existence of the non-linear distortion components and treating the imaging system as if it were just an ordinary linear pinhole camera. Then, the procedure described in Sec. 3.1.1 can be executed, delivering initial values for the intrinsic and extrinsic parameters. Zhang gives a straight-forward approach for determining initial estimates for the radial distortion coefficients in [Zha02]. A subsequent minimization of (3.1.35) gives the final solution for all parameters.

3.2. Omnidirectional cameras

The previous sections covered different aspects of the pinhole camera. First, the model was introduced, then it was shown in Section 3.1.1 how its linear parameters can be identified and finally Section 3.1.2 explained how to describe non-linear lens distortions.
When it comes to wider fields of view, however, the pinhole camera model might not be adequate anymore. This may be due to increased severity of lens distortions, which can not be described sufficiently accurate by the common model. But the main reason is that omnidirectional cameras have the ability to cover more than a hemisphere, i.e. their field of view can exceed 180°. The pinhole model does not cover these cases, as it requires all scene points to lie in front of the optical center.
Omnidirectional cameras can roughly be divided into two classes: fisheyes and catadioptrics. Fisheye cameras use lenses to achieve a wide field of view and can therefore be considered as a natural extension of standard cameras. They are widely used for many different tasks of computer vision. Catadioptrics, on the other hand, use a combination of lenses and mirrors to achieve omnidirectional properties. Their primary goal is to cover panoramic views, which focus on lateral viewing directions, in contrast to fisheyes, which generate mainly frontal views (see Figure 3.2.1 for a visualization). Common applications are mobile robotics and video conferences.

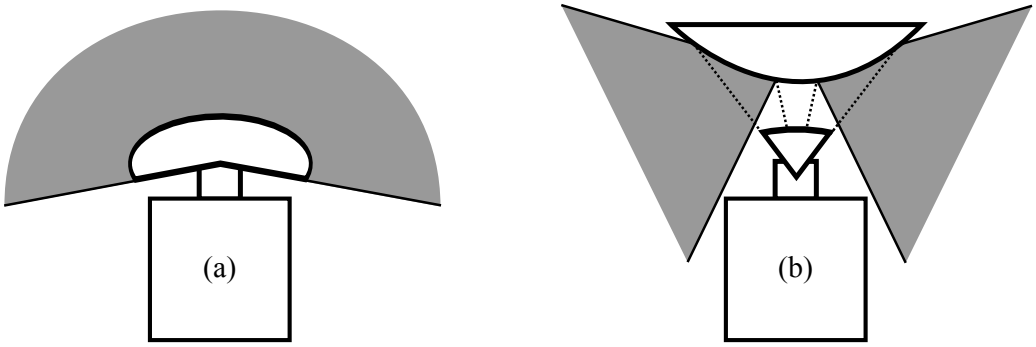


Figure 3.2.1.: Fields of view of a fisheye camera (a) and a catadioptric camera (b).

This section will elaborate on fisheye camera models, considering parameters and calibration in Section 3.2.1. Then, Section 3.2.2 gives a brief overview of catadioptric imaging systems and related models.

3.2.1. Fisheye cameras

In contrast to the distortion models used for pinhole cameras (see Section 3.1.2), fisheye cameras are commonly described by defining a mapping between the angle θ, that is spanned by a viewing ray and the principal axis, and the distance r of the corresponding pixel from the principal point in the image plane. These models are also radially invariant, but unlike the linear projection they are defined for angles bigger than 90°. Formulating the relation for a pinhole camera without non-linear distortions gives

$$r(\theta) = f \tan(\theta/2), \tag{3.2.1}$$

which is called the *stereographic* model. Fisheye lenses, on the other hand, are commonly produced such as they can be described by the linear *equidistant* model, defined as

$$r(\theta) = f\theta. \tag{3.2.2}$$

Further models can be found in [Stu10]. To avoid the difficult task of choosing the correct model for a specific camera, Kannala and Brandt propose in [KB04] to use the polynomial model

$$r(\theta) = k_1\theta + k_2\theta^3 \tag{3.2.3}$$

which is able to approximate the models mentioned before sufficiently well. To accommodate deviations of real cameras from this representation, they introduce further distortion terms in radial and tangential direction. More details on these additional parameters, as well as a calibration method, can be found in their paper.

3.2.1.1. The panomorph camera

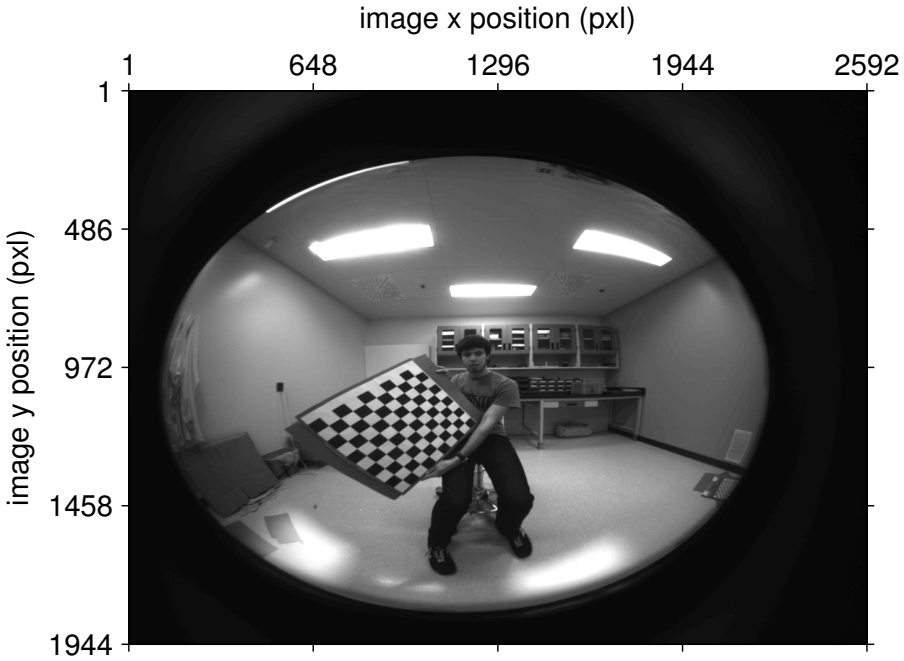
Classical wide-angle fisheye lenses as described before are commonly used for omnidirectional vision. But they have two major disadvantages:

1. As they are rotationally symmetric and the sensors of digital cameras are generally rectangular, a huge part of the image cannot be used for imaging. Therefore, a large percentage of the image pixels is not even used for taking measurements.

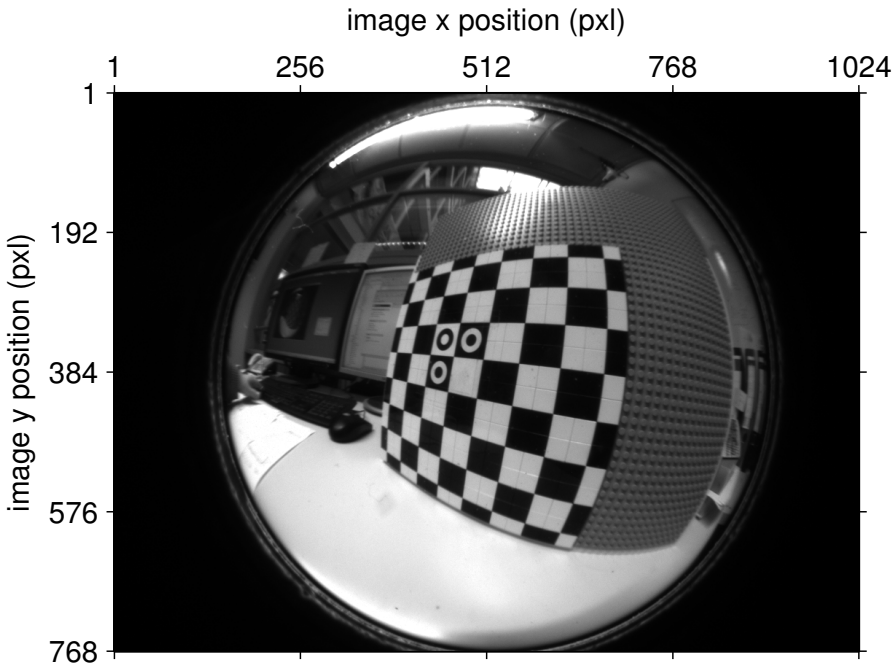
2. The angular resolution decreases with the distance from the optical center. In many applications, however, important information tends to lie in an area that is not in the image center, but closer to the borders.

The company Immervision[1] developed an omnidirectional lens that remedies these drawbacks. Their so-called *panomorph* lens delivers an elliptical image, which fits better on the rectangular sensors of digital cameras. An exemplary camera image can be found in Figure 3.2.2. The field of view is 182° and therefore similar to standard fisheye lenses. To overcome the problem of a decreasing angular resolution with the distance to the principal point, they manage to produce lenses which have a locally varying resolution. This aspect is described in greater detail in Poulin-Girard and Thibault's work [PGT12]. There, the inverse of the so-called *instantaneous field of view* (IFOV) is used to measure the angular resolution. While the IFOV is measured in degrees per pixel, its inverse IFOV^{-1} gives the amount of pixels that cover an angular change of one degree. A lateral profile along the horizontal symmetry axis is shown in Figure 3.2.3 (for a complete two-dimensional profile, please confer [PGT12]). It reveals that the angular resolution is biggest at a distance of approximately 750 pixels from the optical center (which is not the center of the image in this case). Furthermore, as the covered area in the camera image is elliptical and not circular, the distortion is not rotationally symmetric. None of the models described before is able to represent this kind of distortion function. Therefore, a complex setup is proposed in [PGPTD10], which allows to determine the angular resolution function and serves for camera calibration.
The surface model and its calibration procedure proposed in this thesis allows to determine the model parameters in a much more user-friendly and a less hardware intensive way. Corresponding results can be found in Section 6.2.

[1] www.immervisionenables.com



(a) Exemplary image of a panomorph camera.



(b) Exemplary image of a classical fisheye camera.

Figure 3.2.2.: Exemplary images from cameras with different wide-angle lenses.

Figure 3.2.3.: The angular resolution along the main horizontal axis of a camera with a panomorph lens. Values are manually reconstructed from [PGT12].

3.2.1.2. Scaramuzza's omnidirectional camera model

A more general approach for calibrating omnidirectional cameras was proposed by Scaramuzza et al. in [SMS06]. Their model can not only describe fisheye cameras with high accuracy, but also catadioptrics with a single point of view. It will be used for comparison in this work and therefore described in greater detail here. Notations and coordinate systems are adjusted, such as they match the former definitions in this work. The procedure itself, however, is not affected by these changes. The concepts and values described in the following are illustrated in Figure 3.2.4.
The main idea behind Scaramuzza's model is to specify a relationship between the distance of a pixel from the optical center on the image plane and the z component of the direction vector of the corresponding viewing ray. A prerequisite is that the camera to be described has a single point of view, which is also the origin of the camera coordinate system CCS. Its z axis (and therefore the principal axis of the camera) is the rotational axis of the camera lens. The image plane is said to lie parallel to the xy plane of the CCS. Within it, a 2D coordinate system is defined by two axes that are aligned with the x and y axes of the CCS and is centered at the piercing point of the principal axis with the plane. Point coordinates in this metric coordinate system are named as $\boldsymbol{p}_f = (x_f, y_f)^T$. Note that this system is not the coordinate system ICS of the digital image. The relation between $\boldsymbol{p}_f$ in the plane coordinate system and viewing ray direction $\boldsymbol{g}_{cam}$ in camera coordinates is now defined as

$$\boldsymbol{g}_{cam}(x_f, y_f) = \begin{pmatrix} x_f \\ y_f \\ \sum_{i=0}^{N} a_i (r_f)^i \end{pmatrix} \tag{3.2.4}$$

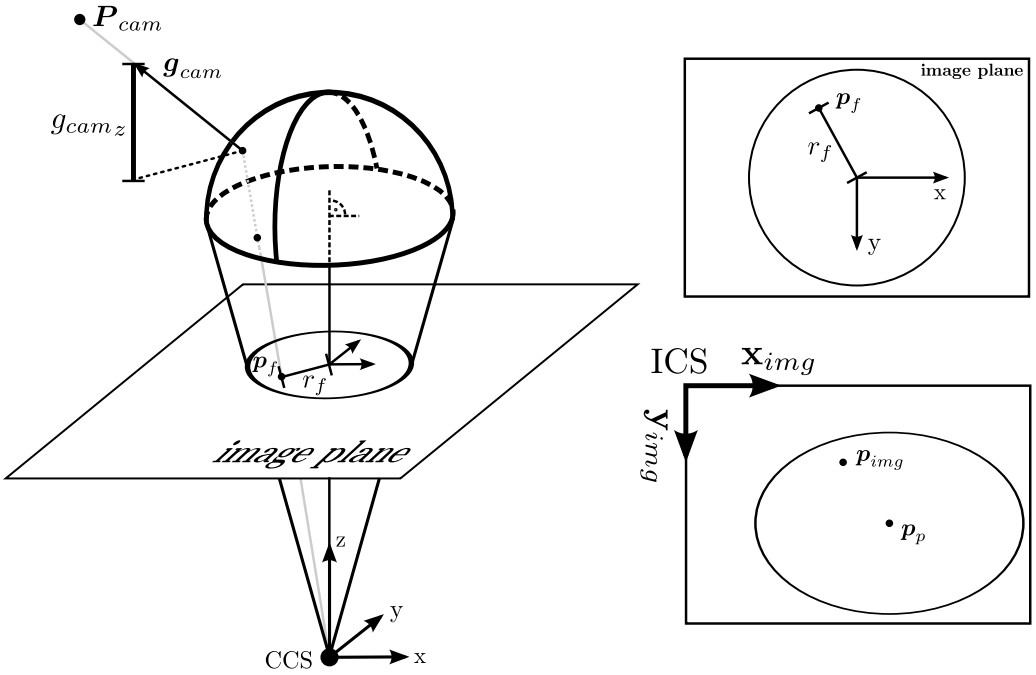


Figure 3.2.4.: Scaramuzza's camera model for a fisheye lens. A point $\boldsymbol{P}_{cam}$ is projected to an image plane point $\boldsymbol{p}_f$ (left). The point in the image plane $\boldsymbol{p}_f$ and the digital image point $\boldsymbol{p}_{img}$ are related as specified in (3.2.5) (right).

with $r_f = ||\boldsymbol{p}_f||$ being the distance from the optical axis. N is the maximal exponent and is usually set to 4. The a_i are the polynomial coefficients, which are the main parameters of the camera model. To determine metric coordinates $\boldsymbol{p}_f$ from pixel coordinates $\boldsymbol{p}_{img}$, an intermediate value $\boldsymbol{p}_u$ is introduced. $\boldsymbol{p}_u$ and $\boldsymbol{p}_{img}$ are related by a combination of an affine transformation A, that compensates for non-quadratic sensor elements, and a shift by the pixel position of the principal point $\boldsymbol{p}_p$. A scaling factor α then connects the intermediate pixel coordinates $\boldsymbol{p}_u$ with the metric coordinates $\boldsymbol{p}_f$. This gives the relation

$$\boldsymbol{p}_f = \alpha \boldsymbol{p}_u = \alpha(\mathrm{A}\boldsymbol{p}_{img} - \boldsymbol{p}_p). \tag{3.2.5}$$

Combining (3.2.4) and (3.2.5) finally provides the camera model equation

$$\boldsymbol{g}_{cam}(\boldsymbol{p}_{img}) = \alpha \begin{pmatrix} \boldsymbol{p}_u \\ \sum_{i=0}^{N} a_i r^i \end{pmatrix}, \tag{3.2.6}$$

where r is $||\boldsymbol{p}_u||$.
Scaramuzza uses hand-held planar chessboard patterns for calibration. Therefore, the plane poses are added to the set of parameters to be determined. They are defined by the (3×4) transformation matrices $\mathrm{M}_v = [\boldsymbol{r}_{v1}\ \boldsymbol{r}_{v2}\ \boldsymbol{r}_{v3}\ \boldsymbol{t}_v]$. Viewing ray direction $\boldsymbol{g}_{cam}$ and a 3D homogeneous point $\mathbf{P}_v$, given in the coordinate system of view v, are connected via

$$\lambda \boldsymbol{g}_{cam} = \lambda\alpha \begin{pmatrix} \boldsymbol{p}_u \\ \sum_{i=0}^{N} a_i r^i \end{pmatrix} = \boldsymbol{P}_{cam} = \mathrm{M}_v \mathbf{P}_v. \tag{3.2.7}$$

Determining the intrinsic parameters $\{a_0, ..., a_N\}$ and the extrinsics of each plane is now the task of the calibration procedure. With the aid of the cross product, (3.2.7) can be rewritten as

$$\boldsymbol{g}_{cam} \times \lambda \boldsymbol{g}_{cam} = \boldsymbol{g}_{cam} \times \mathrm{M}_v \mathbf{P}_v = \mathbf{0}_3 \tag{3.2.8}$$

$$\Rightarrow \quad \begin{pmatrix} x_u \\ y_u \\ \sum_{i=0}^{N} a_i r^i \end{pmatrix} \times [\boldsymbol{r}_{v1}\ \boldsymbol{r}_{v2}\ \boldsymbol{r}_{v3}\ \boldsymbol{t}_v] \begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix} = \mathbf{0}_3. \tag{3.2.9}$$

When assuming that the calibration planes are defined as $z_v = 0$ in the local coordinate systems, (3.2.9) simplifies to

$$\begin{pmatrix} x_u \\ y_u \\ \sum_{i=0}^{N} a_i r^i \end{pmatrix} \times [\boldsymbol{r}_{v1}\ \boldsymbol{r}_{v2}\ \boldsymbol{t}_v] \begin{pmatrix} x_v \\ y_v \\ 1 \end{pmatrix} = \mathbf{0}_3. \tag{3.2.10}$$

This gives three equations (with $f(r) = \sum_{i=0}^{N} a_i r^i$)

$$\begin{aligned} y_u(r_{v31}x_v + r_{v32}y_v + t_{v3}) - f(r)(r_{v21}x_v + r_{v22}y_v + t_{v2}) &= 0 \\ f(r)(r_{v11}x_v + r_{v12}y_v + t_{v1}) - x_u(r_{v31}x_v + r_{v32}y_v + t_{v3}) &= 0 \\ x_u(r_{v21}x_v + r_{v22}y_v + t_{v2}) - y_u(r_{v11}x_v + r_{v12}y_v + t_{v1}) &= 0, \end{aligned} \tag{3.2.11}$$

the last of them being independent of the camera intrinsics and linear in the extrinsic parameters. Stacking those for multiple point correspondences of the same view allows to set up an equation system

$$\mathrm{H}_v \boldsymbol{x}_v = \mathbf{0} \tag{3.2.12}$$

which can be solved for

$$\boldsymbol{x}_v = [r_{v11}\ r_{v21}\ r_{v12}\ r_{v22}\ t_{v1}\ t_{v2}]^T \tag{3.2.13}$$

up to scale in a least-squares sense. r_{v31}, r_{v32} and the correct scale factor can then be determined by exploiting the orthonormality of $\boldsymbol{r}_{v1}$ and $\boldsymbol{r}_{v2}$. Using these values, the other two equations from (3.2.11) become linear functions of the remaining parameters $\boldsymbol{x}_r = (a_0, ..., a_N, t_1, ..., t_{J_v})^T$ with J_v being the number of views used in the procedure. The arising system of equations

$$\mathrm{A}_r \boldsymbol{x}_r = \boldsymbol{b}_r \tag{3.2.14}$$

can be solved directly using the pseudo-inverse of A_r. For details on A_r, $\boldsymbol{b}_r$ and also H_v from (3.2.12), please confer [SMS06]. To identify the best polynomial degree N, Scaramuzza suggests to increase it gradually and repeat the calibration procedure until the reprojection error converges. He points out, however, that $N = 4$ delivers good results, which will therefore be the degree used throughout this work.
During the described procedure, intermediate pixel coordinates $\boldsymbol{p}_u$ are used. To be able

to determine them from pixel coordinates, A and $\boldsymbol{p}_p$ are needed (see (3.2.5)). Scaramuzza proposes to obtain those from the ellipse that marks the actually used image region. In practice, however, it is sufficient to choose $A = I_2$ and $\boldsymbol{p}_p$ as the image center. These and the subsequently determined intrinsic and extrinsic parameters can now serve as starting values for a non-linear optimization procedure that minimizes the reprojection error. The result is the final set of model parameters.
This section showed how fisheye cameras can be described by Scaramuzza's model and how it is calibrated. The next section will elaborate on catadioptric cameras, which use a combination of a camera and a mirror to get an omnidirectional view.

3.2.2. Catadioptric cameras

While the last section showed how lenses can be utilized to achieve a wide field of view, this section will describe another category of cameras which is called *catadioptrics*. These systems make use of a camera with a refracting lens (dioptric) and combine it with a reflecting surface (catoptric), usually with the goal to create an omnidirectional camera. Generally, there are no limitations to the type of lenses used and the shape of the mirror. Very commonly utilized are quadric-shaped mirrors, which are created by rotating conic sections around their symmetry axis. The type of the conic section (parabolic, hyperbolic, conic) has great influence on the characteristics of the imaging system. Many concepts in computer vision and photogrammetry are based on perspective projection. Any central camera allows to generate images that follow the corresponding laws of central projection and therefore permits the application of the related concepts. This holds true even in case the optical center of the camera system is not coincident with the optical center of the involved dioptric camera. To fulfill this requirement of centrality, some care has to be taken when building a catadioptric camera.
There are basically two setups which provide a single effective viewpoint (see also [BN99]):

1. The combination of a telecentric camera (which provides orthographically projected images) with a parabolic mirror.

2. The use of a pinhole camera with a hyperbolic mirror.

The next sections will focus on the geometrical aspects of these setups, covering central as well as non-central cases. An approach for describing these systems can be found in Section 3.2.2.4.

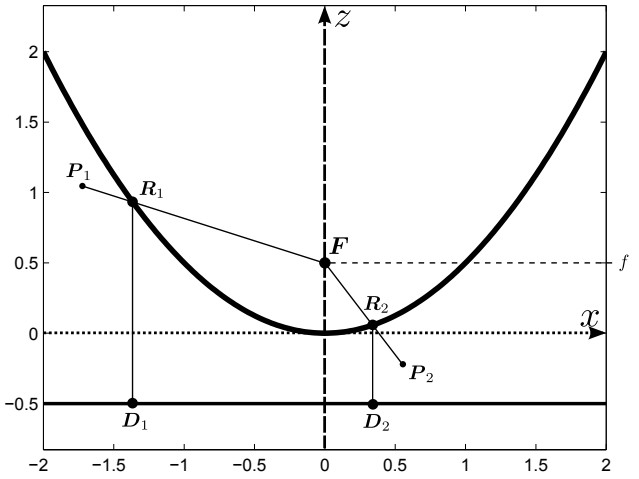


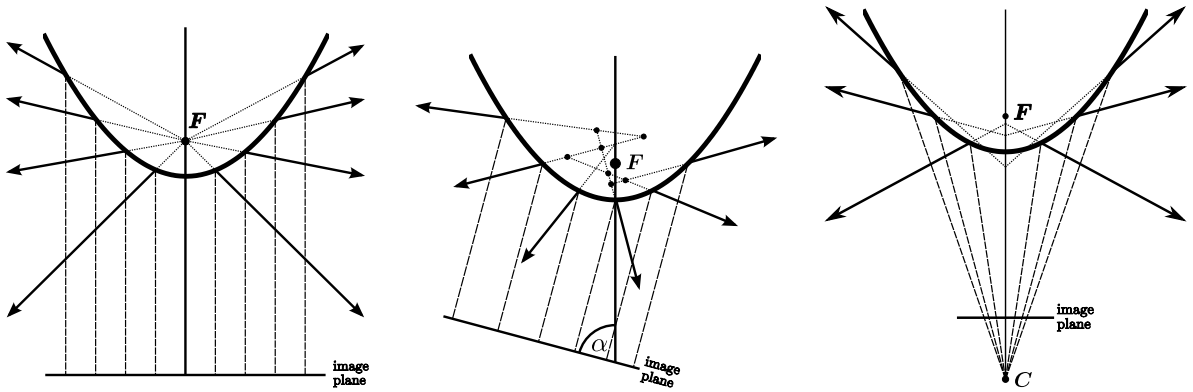
Figure 3.2.5.: Focusing properties of a parabola. The lines $\overline{\boldsymbol{D}_1\boldsymbol{R}_1}$ and $\overline{\boldsymbol{D}_2\boldsymbol{R}_2}$ are parallel to the dashed vertical symmetry axis. They are reflected at the mirror surface at $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ such as the resulting lines $\overline{\boldsymbol{P}_1\boldsymbol{R}_1}$ and $\overline{\boldsymbol{P}_2\boldsymbol{R}_2}$ intersect in the focal point $\boldsymbol{F}$.

3.2.2.1. Paracatadioptric cameras

Paracatadioptric cameras combine a paraboloid mirror with a camera. A paraboloid is generated by rotating a 2D parabola around its symmetry axis. The mathematical formulation is

$$z = \frac{1}{4f}(x^2 + y^2) \tag{3.2.15}$$

where f is the distance of the focal point $\boldsymbol{F}$ from the mirror tip. In this case the rotational axis is the z axis of the coordinate system. A special property of the parabola (and therefore also the paraboloid) is that the reflections of rays that are parallel to the rotational axis meet in the focal point, when extended on the inside of the mirror. Figure 3.2.5 illustrates this. Therefore, a paracatadioptric camera is only central if the dioptric camera is telecentric and the camera image plane is perpendicular to the rotational axis of the mirror. Otherwise, the reflected rays do not intersect in a single point (see Figure 3.2.6). There are two major disadvantages of this setup: first, telecentric cameras are large compared to central ones. Furthermore, the resolution in terms of pixels per angle is biggest close to the rotational axis, which in many applications is the region of least interest.



(a) (b) (c)

Figure 3.2.6.: Paracatadioptric camera setups. Such a system is only central when an orthographic camera perfectly aligned with the mirror is used (a). Misalignment (b) or the use of a central camera (c) lead to non-central configurations.

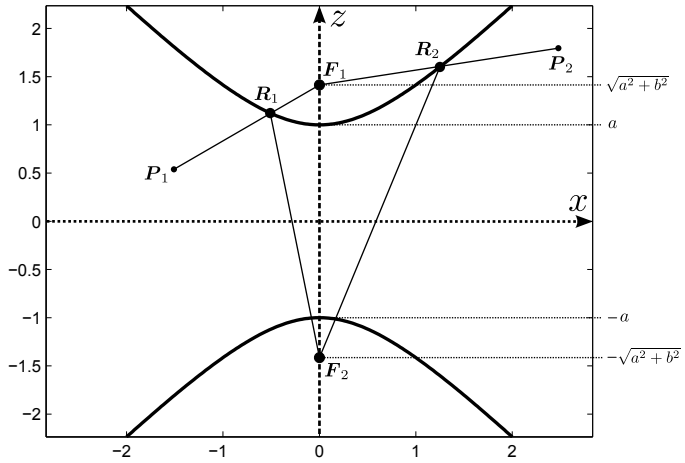


Figure 3.2.7.: A North-South opening hyperbola. Rays emerging from the lower focal point $\boldsymbol{F}_2$ and being reflected in the upper half (e.g. in $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$) converge in the upper focal point $\boldsymbol{F}_1$.

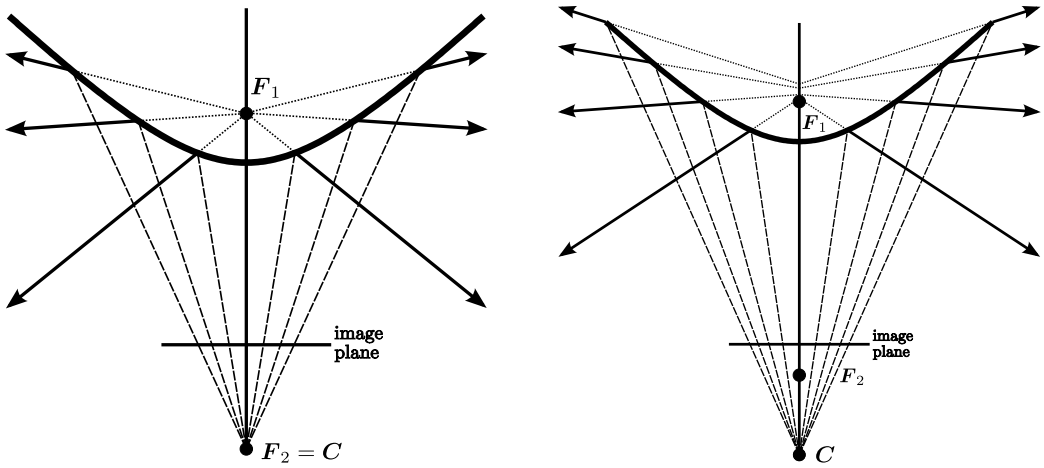


Figure 3.2.8.: Hypercatadioptric cameras can be central (a) or non-central (b), depending on the placement of the perspective camera.

3.2.2.2. Hypercatadioptric cameras

Hyperboloid mirrors are constructed by taking the upper half of a hyperbolic function

$$\frac{y^2}{a^2} - \frac{x^2}{b^2} = 1 \tag{3.2.16}$$

and rotating it around its symmetry axis. An explicit formulation of the hyperbolic function is

$$y = a\sqrt{1 + \frac{x^2}{b^2}}, \tag{3.2.17}$$

which is illustrated in Figure 3.2.7. Such a hyperbola has two focal points $\boldsymbol{F}_{1,2}$ at positions $(0, \pm\sqrt{a^2 + b^2})^T$. Lines starting at $\boldsymbol{F}_2$ in the lower half-plane and being reflected at the graph in the other half-plane lead to rays that intersect in the upper focal point (when extended inside the mirror). Therefore, a hypercatadioptric camera system is only central if the dioptric camera is central, its optical center lies in the second focal point of the mirror and its optical axis is aligned with the mirror's rotational axis. Consequently it is possible to create a central system which is much more compact than a paracatadioptric setup. However, as shown in Figure 3.2.8, translating the camera along the optical axis leads to a non-central setup. Instead, all rays just intersect the central axis. Due to this additional constraint, alignment of mirror and camera is more difficult. Furthermore, the region of high resolution is still in the image center.

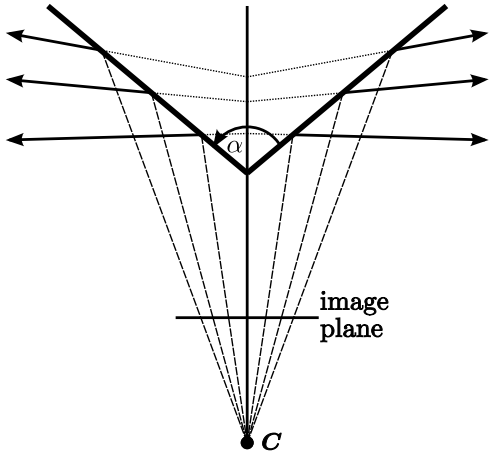


Figure 3.2.9.: An imaging system comprising a conic mirror with opening angle α and a perspective camera.

3.2.2.3. Conic cameras

Conic catadioptrics use a cone-shaped mirror and a central camera to achieve an omnidirectional view. The field of view can be adjusted by changing the opening angle of the cone and the distance of the camera from the mirror. Only one (theoretical) configuration is central, which is when the focal point of the camera lies at the mirror tip. Otherwise, the setup is non-central. To acquire an image with radially symmetric distortions, the camera center has to be placed on the rotational axis of the mirror and the camera's principal axis has to be aligned to it. This setup is shown in Figure 3.2.9.

3.2.2.4. Modeling catadioptric cameras

There are various approaches for modeling catadioptric cameras. The most obvious one is the direct geometric description of the imaging system. In case the intrinsics of the used dioptric camera as well as the characteristics of the mirror and their relative position and orientation are known, ray-tracing can be utilized to realize back projection. The procedure involves the determination of the viewing ray of the utilized perspective camera and its reflection at the mirror surface to get the final viewing ray of the imaging system. Forward projection, however, is a more complicated task which is generally not solvable in closed form. It is therefore often realized by a non-linear optimization procedure.
It is common to focus on imaging systems that have a single effective viewpoint when modeling catadioptric cameras. Those described in Section 3.2.2.1 and Section 3.2.2.2 as well as mirror-less perspective dioptric cameras without non-linear lens distortions can be represented by the *sphere model* introduced by Geyer and Daniilidis in [GD00] and

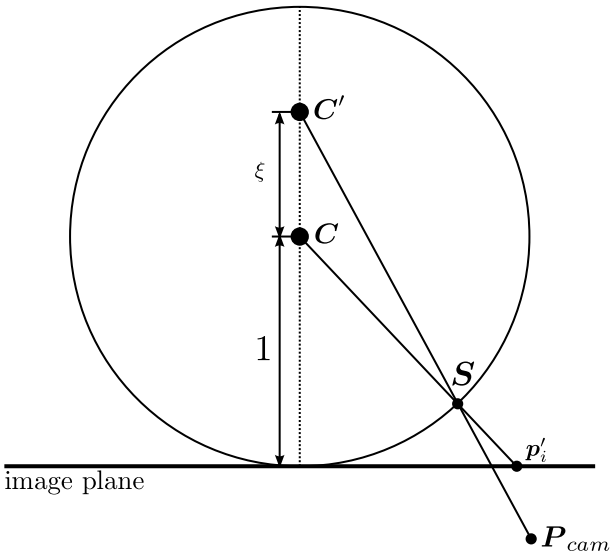


Figure 3.2.10.: The forward projection process of the sphere model works in two stages: first, the point $\boldsymbol{P}_{cam}$ is projected to the unit sphere, then the resulting point $\boldsymbol{S}$ is projected to the image plane. See text for further details.

[GD01]. A visualization can be found in Figure 3.2.10. The model realizes forward projection in a two-step procedure: first, a 3D point $\boldsymbol{P}_{cam}$ is projected to the unit sphere. This is done by intersecting the line which connects $\boldsymbol{P}_{cam}$ and a specified point $\boldsymbol{C}'$ on a selected axis of the sphere with the sphere itself. Second, the resulting point $\boldsymbol{S}$ is projected to a plane that is perpendicular to the chosen sphere axis, using the sphere center $\boldsymbol{C}$ as projection center. This results in a point $\boldsymbol{p}'_i$ in metric 2D plane coordinates which can be transformed to pixel image coordinates $\boldsymbol{p}_i$ with an affine transformation. By changing the distance ξ between $\boldsymbol{C}$ and $\boldsymbol{C}'$, different camera setups can be described, ranging from paracatadioptric ($\xi = 1$) and hypercatadioptric ($1 > \xi > 0$) to perspective ($\xi = 0$) imaging systems.
Also the model proposed by Scaramuzza (see Section 3.2.1.2) can be used to describe central catadioptric systems. The only change to modeling fisheye cameras is the position of the camera coordinate system, which now lies in the actual effective viewpoint inside the mirror and not in the focal point of the dioptric camera (compare Figure 3.2.11 and Figure 3.2.4, respectively).
Countless methods have been proposed for the calibration of models of central catadioptric systems. A very good overview was assembled by Puig et al. in [PBSG11]. They give a list which shows the mirror type, the used model, the utilized calibration pattern and the number of views needed for various calibration procedures. Furthermore, they intensively compare the performance of four different approaches, three of them using the sphere model of Geyer and Daniilidis as a basis and the fourth one utilizing Scaramuzza's model. As it turns out, all of the procedures perform well for most of the analyzed imaging systems. Because Scaramuzza's method performs best for the "unknown-shape catadioptric

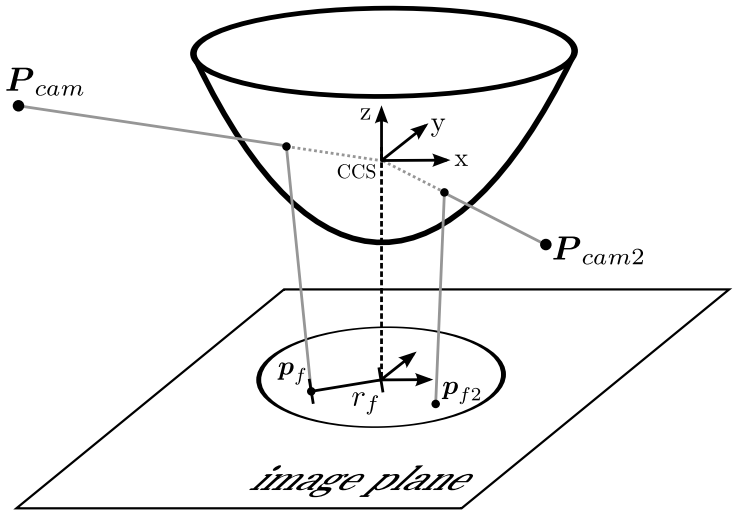


Figure 3.2.11.: Scaramuzza's model for a central catadioptric system. Confer Figure 3.2.4 for comparison with a fisheye camera.

system", which utilizes the mirror also used for experiments in this work (see Figure 6.1.2), this procedure is chosen to generate the comparative results shown later in Chapter 6.
When it comes to non-central systems, the results of Puig et al. indicate that some of the central models and procedures might still be applicable, at least for minor deviations. Also other works use the central assumption, although their systems do evidently not possess a single effective viewpoint, e.g. [DK00] or [MP04], where a spherical mirror is used. The alternative to these approximating assumptions is to create an exact geometrical model of the setup. In this case it is convenient to describe the *caustic* of a camera, which is a term introduced by Born and Wolf in [BW70]. It can be described as the smallest possible region in 3D space that is tangent to all viewing rays. Swaminathan et al. state in [SGN06] that "caustics completely describe the geometry of an imaging system", as each pixel is mapped to a point on the caustic, which directly defines the starting point of the corresponding viewing ray and also its direction via the tangent. Their concept is especially useful for systems which have a rotational symmetry, e.g. catadioptrics where the principal axis of the perspective camera is aligned with the symmetry axis of the mirror. In that case it is possible to work on cross-sections of the mirror which simplifies the application of their approach. They deliver algebraic descriptions for caustics of various setups in [SGN01] and [SGN06] and also propose a procedure to determine them from known camera motion. An approach for more general systems, e.g. with unaligned axes, is not suggested.
The methods mentioned before all describe the process of back projection. Its inverse, the forward projection, is usually not considered. There are some works which use iterative approaches to solve the problem for systems with known geometry, e.g. [MP04] or [Lhu08]. Agrawal et al. deliver an analytical solution, at first for axial cameras in [ATR10]. They show in [ATR11] that the concept also works with quadric mirrors and perspective cam-

eras in general position. However, the geometry of the setup has to be exactly known and the procedure involves the determination of the roots of an 8th degree polynomial. They state that it is possible to identify the correct solution by applying Snell's law, but do not elaborate on that.
This section showed various omnidirectional cameras and introduced models to describe them along with selected calibration procedures. It becomes more and more obvious that camera calibration demands a lot of insight and technical expertise. The next section therefore introduces concepts which are more general and therefore supposedly simplify the task of camera calibration.

3.3. Generic camera models

As the last section showed, there are many different kinds of cameras and corresponding models (and many more can be found in Sturm's survey [Stu10]). This makes camera calibration a difficult task, because the user has to know the structure of the system and also has to choose the appropriate calibration procedure. When it comes to non-central catadioptric systems, the solutions are rather specific, which makes calibrating and using the camera even more cumbersome.
For these reasons it might be convenient to have a model that is not specific for a certain type of camera, but generic in a sense that multiple different systems can be described equally well by it. Although this promises to make calibration and usage of cameras much easier, there are in fact just a few publications which aim in that direction. Here, at first the so-called *two-plane model* will be introduced, which is proposed by Chen in [CBK80] and is the first attempt to create a single model for different kinds of imaging systems. However, as it has its drawbacks, the remaining sections will focus on the *generic camera model* proposed by Grossberg and Nayar in [GN01], which is more general and offers an even greater flexibility.

3.3.1. The two-plane model (Chen et al.)

Chen et al. introduce the idea of a two-plane model in [CBK80]. They state that, against a common assumption, cameras are possibly non-central in reality and therefore propose a more generic concept for camera modeling. It involves taking measurements for a specific image pixel on two fixed planes in space and storing the corresponding positions. These positions serve to determine the viewing ray for that pixel. They propose to use a robot arm that positions a point light source within these planes for calibration. This procedure is executed for only a fraction of the image pixels and intermediate positions

are calculated with an interpolation method. Experiments prove that their approach is adequate to get accurate viewing rays for arbitrary pixel positions.
This idea is further elaborated on by Martins et al. from the same research team in [MBK81]. They suggest interpolation functions that are either linear or quadratic in the pixel positions and mainly regard the case that both of the planes are parallel to the image plane, as this reduces the amount of parameters needed. Experiments are conducted for three different cameras and reveal that a linear spline interpolation function delivers the best results. The model is explicitly used for back projection, forward projection is considered to be "impractical". Gremban et al. [GTK88] also apply linear spline interpolation and propose to realize forward projection by using a pinhole camera model to approximate the camera mapping (and therefore ignore non-central aspects). They also suggest to do this locally, i.e. describe the camera by various separate pinhole models for different image regions. A slight increase in model accuracy is detected, but the concept is only applied to cameras with minor lens distortions and therefore leaves the question unanswered of how non-linear aspects influence the results.
Wei and Ma extend the two-plane approach in [WM91] by proposing to use planes at known but arbitrary relative positions and cubic rational functions of the pixel coordinates for interpolation. They also only consider back projection. Their follow-up publication [WM93] proves that the model can be used to describe perspective cameras and pinhole cameras with minor distortions. Furthermore, they propose a method to realize forward projection, but only for imaging systems with a single center of projection.
The idea of using two planes for camera modeling is also investigated by Champleboux et al. in [CLSC92]. They enhance the interpolation concept by using B-spline surfaces and propose to use more than two planes (with known relative poses) to increase the model accuracy. Forward projection is not mentioned in that publication.
To sum up the contributions concerning the two-plane model: the concept itself has the ability to describe central as well as non-central imaging systems, though it was not yet used to model a clearly non-central setup. Non-linear distortions can be handled by choosing an appropriate interpolation function. One major drawback is the lack of a formulation for general forward projection. Furthermore, as all viewing rays have to intersect two planes, the opening angle of the field of view cannot exceed 180°. This prevents the description of omnidirectional cameras. Another concept for general camera modeling which does not have this disadvantage will be introduced in the next section.

3.3.2. The generic model (Grossberg and Nayar)

The field of view that can be covered with the two plane model described in the last section is restricted by the size of the planes. Therefore, it does not allow to describe omnidirectional cameras. This section will elaborate on the *generic camera model*, which was first proposed by Grossberg and Nayar in [GN01]. It has the potential to describe any

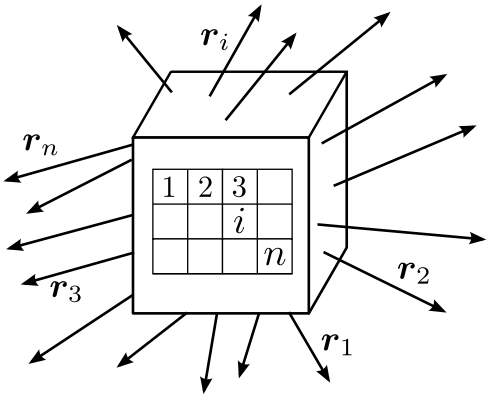


Figure 3.3.1.: The discrete camera model proposed by Grossberg and Nayar combines each image pixel i with an arbitrary and independent viewing ray $\boldsymbol{r}_i$ to form a *raxel*.

camera system imaginable and can therefore rightfully be called *generic*. First, the main idea and related concepts of the inventors will be introduced. Later on in this section, the works of other research groups utilizing this model will be presented.
Grossberg and Nayar propose to consider an imaging system as a black box. They assume that each pixel of the digital image is connected to an arbitrary viewing ray. There are no physical laws or analytical descriptions that describe this connection, which reduces the model to a simple look-up table that stores viewing ray parameters for every single pixel. Grossberg and Nayar call this combination of ray and pixel a *raxel* which emphasizes the unity of the two entities. Figure 3.3.1 visualizes this concept. In fact, their definition of a raxel does not only include geometrical information, such as pixel position, ray position and ray direction, but also incorporates radiometric parameters. In this work, however, only the geometric aspects will be considered. Following the terminology of other works, the respective model is called the *generic camera model*.
Along with the abandonment of a mathematical relation between image position and ray parameters comes the capability to assign completely independent rays to all the pixels. This provides the possibility to describe central as well as non-central camera systems and even cameras with discontinuities, e.g. a conic catadioptric, or multi-camera systems. Therefore, it can be used to describe any camera and eliminates the need for model selection. The practical applicability of the generic camera model has been shown by utilizing it for classical tasks of computer vision, such as pose estimation and 3D reconstruction [Ple03] [SRL05].
This great flexibility of the generic camera model comes with a price, though: its discreteness results in a time-consuming calibration procedure which may also demand the utilization of high-precision 3D calibration patterns. Basically, at least two measurements in 3D space are needed for each pixel, allowing to determine the corresponding viewing ray. Grossberg and Nayar [GN01] propose to solve this task by using two views of a plane which is translated by a known distance while keeping its orientation. To identify the local

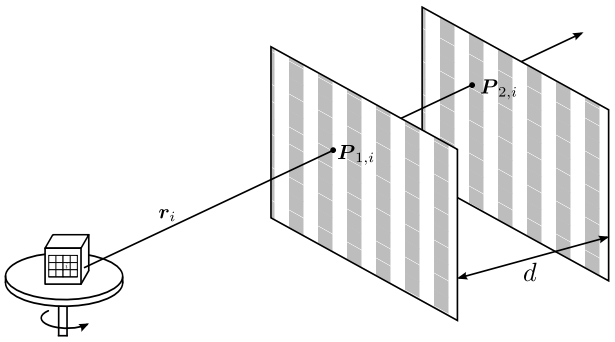


Figure 3.3.2.: The calibration setup proposed by Grossberg and Nayar: a plane is displaced by the known distance d to get two 3D measurements $\boldsymbol{P}_{1,i}$ and $\boldsymbol{P}_{2,i}$ for each ray. The camera can be rotated to expand the calibrated region of the image.

positions on the planes they use a coded light approach. With this static approach, only that region of the image which is covered by both calibration planes can be calibrated. Depending on the camera and its field of view, this region is possibly only a small fraction of the complete image. To expand the modeled area, Grossberg and Nayar rotate the camera on a turntable to calibrate different regions in the image. Figure 3.3.2 illustrates the setup. As this procedure requires the translation of a plane with high accuracy and the implementation of a coded light approach, a high level of expertise is needed to execute it. Furthermore, it is questionable if a rotation of the camera around a single axis suffices to place the calibration planes in all regions of the camera image.
Without any analytical relation between neighboring pixels, general back projection, i.e. the determination of ray parameters for subpixel positions, is not directly possible with the generic camera model. But this is a prerequisite for executing high-accuracy measurement tasks. As Ramalingam points out in [RLS04], the discrete model is not adequate when solving a classical bundle-adjustment task because the underlying non-linear optimization algorithms (e.g. Levenberg-Marquardt) need a continuous cost function to succeed.
By definition, the generic camera model provides back projection because the viewing ray for each pixel can directly be obtained. Forward projection, on the other hand, is not as easily accomplished. As long as a 3D point does not lie exactly on one of the known viewing rays, the corresponding image position can not be determined. These two aspects of subpixel back projection and general forward projection motivate the formulation of a continuous description which introduces interpolation capabilities.
To get a more consistent representation of the generic camera model, Grossberg and Nayar introduce a so-called *ray surface* in [GN01], which is a surface in 3D space where all viewing rays have their starting points (e.g. a sphere). They claim that the caustic of an imaging system is a good choice for the ray surface and propose a general method to determine it in [GN05]. Their suggested *caustic raxel model* is a continuous and piecewise differentiable description of the mapping from image position to ray parameters. They use two mappings to two different planes for each pixel, a concept which is very similar to

the two-plane model of Chen et al. The representation is therefore able to describe central as well as non-central setups, but can not be used for cameras with a field of view that exceeds 180°. A suggestion of how to solve the problem of forward projection is not given. As mentioned before, other research groups saw the potential of the generic camera model and started working with it. These works basically try to remedy the drawbacks of the model in its original form: the complexity of the calibration procedure and the reduced practicability due to its discreteness and therefore missing general solutions for subpixel back projection and general forward projection. The next sections will give an overview of the most important works.

3.3.3. Generic calibration (Sturm and Ramalingam)

Sturm and Ramalingam use hand-held calibration planes in [SR04] to simplify the calibration process, following the tradition of Zhang [Zha02] for *flexible camera calibration*. As their calibration pattern is a sparse grid of features, not every pixel has a measurement for all calibration images. They assume a certain continuity of the camera mapping and use homographic interpolation to identify local positions on each calibration plane for all pixels. Collinearity constraints which state that all measurements for a single pixel need to lie on the same line are formulated, and then exploited to determine the plane poses and hence the parameters of the viewing rays. The main drawback of their procedure is that not the complete image is calibrated and that there are different approaches for central and for non-central camera systems. Using the non-central approach to calibrate central cameras leads to singularities and renders the constructed equation systems unsolvable. Therefore, it has to be known beforehand whether a camera system is central or not. It is proposed in [RSL05] to use a rank analysis of the involved matrices to make this decision automatically, an idea that seemingly has not been put into practice yet. In a follow-up work by Ramalingam et al. [RSL05] the procedure is extended to calibrate further areas of the camera image. This process will be called *complete calibration* in this thesis. The idea is to iteratively add further planes and determine their poses using already known camera rays (see Figure 3.3.3 for an illustration). The procedure involves solving an 8th degree polynomial. It is very sensitive to noise, as only the minimum amount of three rays is used to get a solution. Integrating further information to get a more robust result is not directly possible with the proposed method. Therefore, a RANSAC-based approach is used to identify good solutions.
Apart from the pose estimation process, a bundle adjustment procedure is introduced that minimizes the distance between viewing rays and 3D plane points to refine the plane poses as well as the ray parameters. This allows to also calibrate “slightly non-central” cameras after getting initial solutions with the central algorithm.

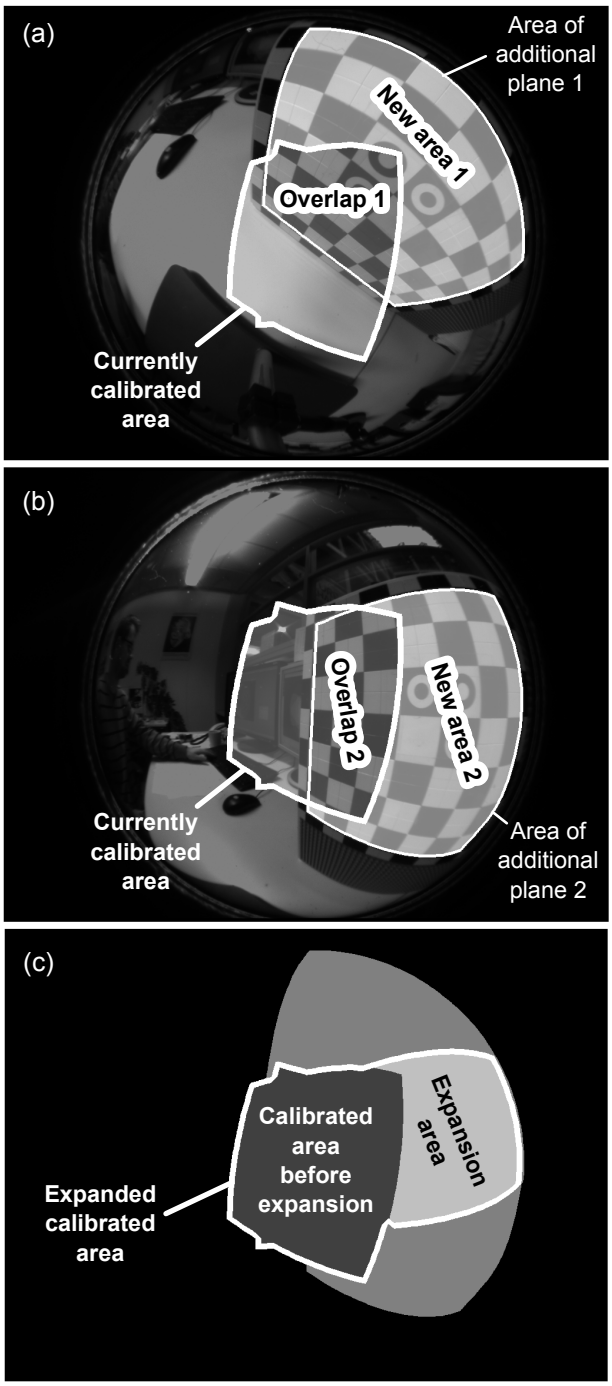


Figure 3.3.3.: Adding further views serves to expand the calibrated image region. (a),(b): Starting from the currently calibrated region, the pose of additional calibration planes can be determined from rays in their overlapping areas. (c): If these additional planes cover a common region in the image, new viewing rays can be determined for that area which expands the calibrated area (images taken from [RW12a]).

3.3.4. Generic central calibration (Dunne, Mallon and Whelan)

Dunne et al. also make use of the generic camera model. In [DMW10] they compare its performance to Zhang's model [Zha02] for three different central cameras. For calibrating the generic model they use a combination of the original approach of Grossberg and Nayar [GN01] and the one proposed by Sturm and Ramalingam [SR04]: calibration planes are placed arbitrarily, but local plane positions are identified with the help of a coded light approach. Instead of homographic interpolation, they use electronic displays to show the patterns and call them *active grids*. Subsequently, a bundle adjustment procedure is executed to refine the results. Their conclusion is that the results are similar to those obtained by Zhang's method for cameras with minor distortions, but that the generic model performs much better for systems with severe distortions.
Later on, the same team developed a new calibration procedure, which gets a solution for the plane poses by exploiting the assumption of centrality and therefore avoids the use of Sturm's and Ramalingam's approach. The same concept will be used in this work to determine an initial solution, which is why it will be described in further detail in the next section.

3.3.4.1. Linear initial calibration

For central catadioptric cameras, it is possible to undistort the images to generate perspective views. This can be done by choosing a *virtual image plane*. Intersecting viewing rays with this plane gives local positions that can be interpreted as locations in an image of a perspective camera. In case a single point of view exists, this procedure is nothing else but a perspective projection. With this approach, images free of non-linear distortions can be generated. This is done in practice by selecting one of the V calibration planes to be the virtual image plane. Without loss of generality the number $v = 1$ is assigned to the chosen plane. The selected plane preferably covers a big area in the original camera image, as it marks the region where the virtual camera can actually see. Undistortion can now be realized by determining the local plane positions $\boldsymbol{p}_{1i}$ for every pixel i in that area (Dunne et al. use their active grids for this step), and reinterpreting them as positions in the virtual image. Figure 3.3.4 illustrates the linearization process for a paracatadioptric imaging system. In Figure 3.3.5 the corresponding new virtual camera with its coordinate system, intrinsic and extrinsic parameters and an exemplary viewing ray are visualized.
By generating a look-up table that contains plane positions for every pixel in the covered area, a complete rectification of any camera image can be realized. The intensity values of the original image define the colors of the corresponding positions of the virtual image. For camera calibration, however, it is not necessary to generate complete images. Only the locations of the points needed for calibration (e.g. chessboard corners) have to be determined in the virtual image. Then, standard linear methods can be used to calibrate

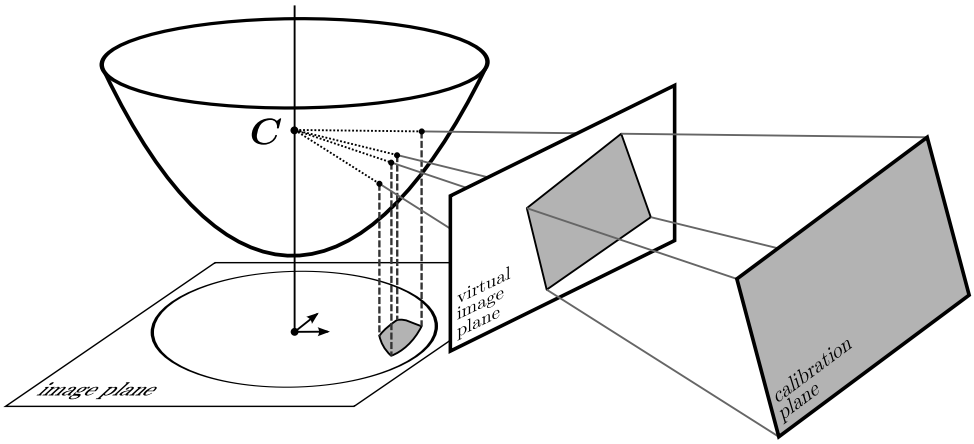


Figure 3.3.4.: To achieve linear calibration, one calibration plane is chosen as a virtual image plane. The actual optical center $\boldsymbol{C}$ of the catadioptric system is now the center of projection of the new virtual camera, which is described in further detail in Figure 3.3.5.

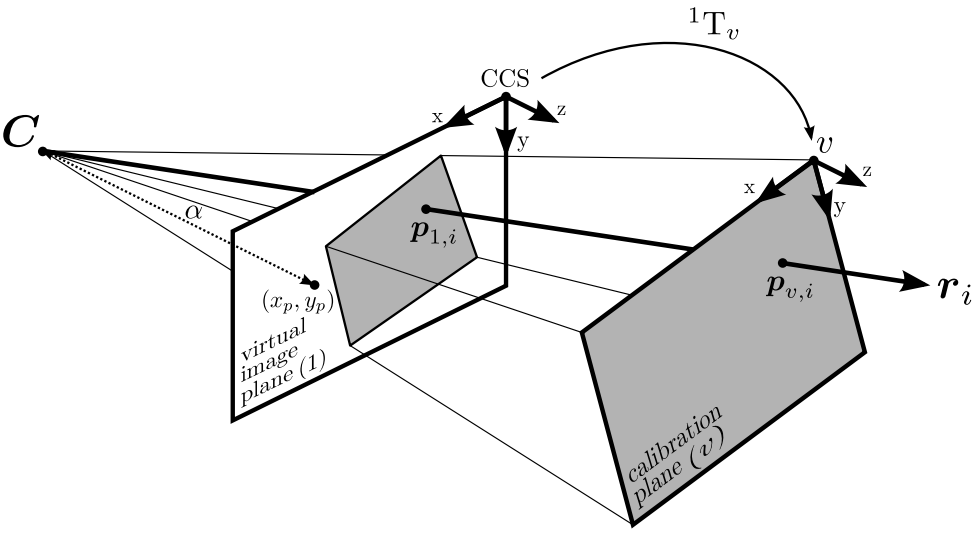


Figure 3.3.5.: The virtual camera generated from the catadioptric system in Figure 3.3.4. $\boldsymbol{C}$ is still the optical center, calibration plane 1 is chosen to be the virtual image plane and defines the camera coordinate system CCS. Viewing ray $\boldsymbol{r}_i$ intersects plane 1 at local position $\boldsymbol{p}_{1,i}$ and plane v at $\boldsymbol{p}_{v,i}$. Focal length α, principal point $(x_p, y_p)^T$ and extrinsic parameters ${}^1\mathrm{T}_v$ are determined by a linear calibration procedure.

the virtual camera, e.g. the one described in Section 3.1.1. A prerequisite is the existence of a virtual plane which shares its image region with at least two other calibration planes. With this requirement fulfilled, the mentioned procedure can be used to determine the principal point $(x_p, y_p)^T$ from (3.1.27) and (3.1.28), the projection coefficient α from (3.1.29) and the extrinsics $\boldsymbol{r}_{1v}$, $\boldsymbol{r}_{2v}$, $\boldsymbol{r}_{3v}$ and $\boldsymbol{t}_v$ of plane v from (3.1.30) and (3.1.31). Here, the image points are measured on the virtual image plane and are therefore given in metric coordinates and not in pixel coordinates. Consequently, the results for x_p, y_p are also in metric coordinates. Furthermore, the pixel scaling factor s is equal to 1, which makes α the actual focal length. The camera coordinate system is now said to be the one of the virtual image plane which leads to the coordinates of the projection center

$$\boldsymbol{C} = \begin{pmatrix} x_p \\ y_p \\ -\alpha \end{pmatrix}. \tag{3.3.1}$$

As the camera coordinate system no longer lies in $\boldsymbol{C}$, the poses of the other calibration planes, need to be adjusted. With $\mathrm{R}_v = [\boldsymbol{r}_{1v} \quad \boldsymbol{r}_{2v} \quad \boldsymbol{r}_{3v}]$ and $\boldsymbol{t}'_v = \boldsymbol{t}_v + \boldsymbol{C}$:

$${}^{cam}\mathrm{T}_v = \begin{bmatrix} \mathrm{R}_v & \boldsymbol{t}'_v \\ \boldsymbol{0}^T & 1 \end{bmatrix}. \tag{3.3.2}$$

This concludes the initial calibration procedure. Ray parameters can be determined for every pixel i within the image area covered by the virtual image plane by fitting a line through camera center $\boldsymbol{C}$ and plane positions $\mathbf{P}_{1,vi} = {}^1\mathrm{T}_v\mathbf{P}_{vi}$, $v \in [2; V]$, in the camera coordinate system. Dunne et al. use an iterative bundle adjustment procedure to refine the camera center as well as the plane poses. Each iteration consists of two steps:

1. The ray parameters $\boldsymbol{r}_i$ are determined for selected pixels for the line defined by $\boldsymbol{C}$ and the mean position of all $\boldsymbol{P}_{1,vi}$ which are calculated using the current values of ${}^1\mathrm{T}_v$.

2. The summed *ray-point distance* between the rays and the corresponding $\boldsymbol{P}_{1,vi}$ is minimized by a non-linear optimization procedure with $\boldsymbol{C}$ and ${}^1\mathrm{T}_v$, $v \in [2; V]$, being the parameters (the virtual image plane with $v = 1$ stays fixed to get a stable optimization procedure). If the procedure has not converged (i.e. if there is still a big change in the parameters), both steps are repeated with the adjusted values for $\boldsymbol{C}$ and ${}^1\mathrm{T}_v$.

After the bundle adjustment procedure, new planes, which were not used for the initial calibration, are added to expand the calibrated image area. The proposed approach for estimating the plane pose again takes advantage of the centrality assumption. It includes the formulation of a linear least squares problem, which has a single solution and is therefore much more simple to use than Ramalingam's method.

3.3.5. The smooth model (Miraldo and Araujo)

The works of Sturm (Section 3.3.3) and Dunne (Section 3.3.4) are exploring different ways to calibrate the generic camera model presented by Grossberg and Nayar. But eventually they keep its discrete form. Consequently, they do not propose any solutions to the problems of subpixel back projection and general forward projection, which are helpful or sometimes even necessary when using the model in practical applications.
Miraldo and Araujo take a different approach in their works. In [MA10] they suggest to use *radial basis functions* as a continuous representation of the generic camera model, assuming that the camera mapping from pixels to rays varies *smoothly*. This is why they call their representation a *smooth camera model*. They choose Plücker coordinates to parameterize the rays. This results in a 6D continuous model description $\mathbf{s}(\boldsymbol{x}) : \mathbb{R}^2 \to \mathbb{R}^6$ where the function parameters $\boldsymbol{x}$ are the pixel coordinates. Their experiments show that already 200 data points are sufficient to get a good approximation of a non-central catadioptric. This dramatically reduces the amount of model parameters (compared to the discrete case) and furthermore provides the ability of subpixel back projection. Although a continuous representation of the generic camera model is now given, they do not deliver an approach for general forward projection.
Subsequently, Miraldo and Araujo published works on how to use radial basis functions to directly calibrate the generic camera model, instead of only using them for representational purposes. In [MA11] it is shown how the parameters of the continuous model $\mathbf{s} : \mathbb{R}^2 \to \mathbb{R}^6$ can be determined by taking 3D points as input data points, instead of 6D Plücker line vectors. It is proven in [MAQ11] that the solution for the parameters is unique which verifies that the measurement of only one 3D point for each pixel is sufficient to parameterize the smooth model. Experiments conducted in [MA11] with simulated and real data prove that the resulting continuous description serves as a good representation of the regarded imaging systems. The real input data was obtained by taking images of calibration planes with chessboard patterns and determining the 6D pose of these planes with an external tracking system. Other than the approaches of Grossberg and Nayar, Sturm and Ramalingam and Dunne et al., Miraldo and Araujo do not rely on a multiple-step procedure which starts by calibrating a small part of the image first and expanding it later on. Instead, they take measurements all over the camera image and calibrate the complete area in a single step. The performance is evaluated by determining the parameters of the smooth model with real input data and measuring the distance between externally determined 3D positions of a test object and the corresponding viewing rays. The mean error ranges from approximately 0.5cm to approximately 3cm, depending on the camera and the type of basis functions used. The distance from the camera at which these measurements were taken is not mentioned, but can be assumed to be within a few meters. In [MA13] the concept is applied to non-smooth systems, such as multi-camera setups. This basically works, but introduces big errors at the discontinuities.

The current chapter introduced the basic concepts of camera modeling and calibration, starting with the pinhole model, continuing with omnidirectional cameras such as fisheyes and catadioptrics and finally explaining the abilities of generic models. The generic camera model of Grossberg and Nayar is widely used, but has some severe drawbacks due to its discreteness. How the so-called surface model as a new continuous representation can be used to overcome these drawbacks is one of the main contributions of this work and will be shown in the next chapter.

Chapter 4

The surface model

In the last chapter, various camera models and selected calibration procedures to determine their parameters were regarded. This included the standard pinhole model as well as omnidirectional models for fisheye cameras and catadioptric systems. Furthermore, two generic camera models were presented, namely the two-plane model and the generic raxel camera model proposed by Grossberg and Nayar [GN01]. It was shown that those are much more versatile and allow to describe non-central setups as well. A major disadvantage of these models is that, due to their discreteness, they lack basic functions which are necessary to use them for computer vision purposes. This chapter will introduce a new continuous representation of the generic camera model. It is called the *surface model* and provides the abilities of subpixel back projection and general forward projection. Besides, the aspect of uncertainty will be integrated into the model, allowing to obtain covariance matrices in addition to the resulting ray parameters and pixel positions of these procedures.

Traditional camera descriptions like the pinhole model with lens distortions (Section 3.1.2) or Scaramuzza's omnidirectional model (Section 3.2.1.2) make use of a restricted set of parameters to define model functions that specify the forward and/or back projection procedure. Therefore, they are often called *parametric*. The generic camera model from Grossberg and Nayar (Section 3.3.2), on the other hand, is commonly said to be *non-parametric*, as no such algebraic description exists. This is misleading, because there is also a set of parameters, which is just much bigger than for the traditional models. For this reason, Sturm proposes in [Stu10] to use the terms *global model* and *local model* instead. Traditional models would be global, as they have a single model function that is valid for every pixel of the camera image. Local models, on the other hand, have descriptions which are exclusive for certain regions of the image. Said differently: forward and back projection do not everywhere depend on the same set of parameters. In this sense, the generic raxel

model is definitely local. It could be said that it is the most local model possible, as it has a different set of parameters for every single pixel. However, it could also be argued that it is not local enough, as it does not provide any information for subpixel positions. Continuous descriptions were already introduced for the two-plane model (Section 3.3.1), the main intention at that time being that not enough storage space was available to save the parameters of every pixel. Those models are defined piecewise and can be considered as being local. Continuous descriptions definitely have their merits, such as reduction of the number of parameters and easy subpixel back projection. Nevertheless, the smooth model proposed by Miraldo and Araujo is the only representation of that kind for the generic camera model (Section 3.3.5). However, the authors did not suggest a method to realize general forward projection.
One aspect that is often neglected when it comes to camera models and their calibration is the issue of uncertainty. Neither the presented classical global models nor the local ones take into account, that no parameterization can be indefinitely accurate. This introduces deviations of the chosen representation from the real camera and inevitably influences the quality of taken measurements. It is by all means beneficial for the user of a camera model to get information on the involved uncertainties, as they can be used in subsequent procedures and provide a valuable basis for assessing the results.
The main goal of this section is therefore to introduce an uncertain spline surface as a continuous description for the generic camera model which provides methods for subpixel back projection and general forward projection. This covers central and non-central systems as well as cameras with locally differing distortions. It is assumed that the camera mapping is locally continuous, justifying the use of an interpolation function. Furthermore, the camera is supposed to have no defocus blur, i.e. the imaging process is independent of an object's distance from the camera. Uncertainties will be provided in both directions of the camera mapping: back projection delivers ray parameters and their covariance matrix, giving uncertain lines in 3D space. The result of the forward projection of an uncertain 3D point to the image will be a 2D pixel position with the corresponding uncertainty in the image plane. Table 4.0.1 shows an overview of the different model representations and their properties, where the last row illustrates that the representation which will be introduced here is the most powerful one. The following sections will elaborate on different aspects of the model, starting with the description of the construction of the spline surface itself in Section 4.1. Section 4.2 will subsequently introduce and evaluate subpixel back projection. General forward projection is then described in Section 4.3. The aspect of uncertainty for both projection directions is covered in Sections 4.2.3 and 4.3.1, respectively.

Authors	Representation	C	SBP	GFP	U
Grossberg and Nayar 3.3.2	discrete raxels				
Sturm and Ramalingam 3.3.3	discrete raxels				
Dunne et al. 3.3.4	discrete raxels				
Miraldo and Araujo 3.3.5	radial basis functions	x	x		
Rosebrock and Wahl [RW12b, RW12a]	two 3D spline surfaces	x	x	x	
this work	single 6D spline surface	x	x	x	x

Table 4.0.1.: Comparison of the properties of the representations of the generic camera model used by different research groups (C=continuous, SBP=subpixel back projection, GFP=general forward projection, U=uncertainties).

4.1. A spline surface in 6D Plücker space

The desirable properties of a representation of the generic camera model are:

1. continuity, providing interpolation abilities
2. flexibility, allowing to describe a wide range of different setups with local variations
3. scalability, facilitating the adaption of the number of parameters
4. uncertainty, offering the ability to handle measurement noise and providing the user with uncertainty information.

To this end, uniformly parameterized B-spline surfaces are chosen (confer Section 2.3 for the mathematical description). They are continuous mappings that offer a great flexibility in terms of modeling capabilities. The surface parameters u and v represent the 2 axes in the image plane. They lie within the range $[0; 1]$ and are linearly dependent on the actual pixel positions due to the uniform spline parameterization. By choosing the number of control points, the amount of parameters can be adjusted such as specified requirements are met. In addition to that, spline surfaces offer the possibility to easily incorporate measurement uncertainties, the main reason being that the relation between data points and control points is linear. Therefore, spline surfaces have all the desired properties and will be used for representing the generic camera model. This approach was also taken by Rosebrock and Wahl in [RW12b] and [RW12a].

The next question to be answered is which line representation is to be used. For traditional central camera models, all viewing rays intersect in a single point. Therefore it would be sufficient to store this optical center and for each ray only the direction vector. Non-central cameras, however, do not possess a single common point of view. Consequently, all lines have to be stored with a full set of parameters.
A line in 3D space has four degrees of freedom, therefore a minimal representation has four parameters. Minimal representations, however, are commonly not free of singularities, which makes their use for omnidirectional cameras cumbersome. In [RW12b] each viewing ray was defined by a starting point and a direction vector. There, the continuous description comprises two separate spline surfaces $\boldsymbol{S}_s(u, v)$ and $\boldsymbol{S}_r(u, v)$ in 3D space, modeling starting points and direction vectors, respectively. The values of these surfaces immediately deliver the viewing ray parameters for any parameter position, without being restricted to integer numbered pixels. This solves the problem of subpixel back projection.
The modeling accuracy of spline surfaces with uniform parameterization, i.e. equidistant knots, is best when the data points are equidistant. For this reason, direction vectors could be normalized and starting points placed on a sphere, as proposed in [RW12a]. A remaining problem is the definition of the center position and the radius of this sphere. For the position, a point that will be called the *caustic center* would be a good choice. It is the point which has minimum distance to all viewing rays and can be determined by using the approach from Section 2.4.4.1 which calculates the best intersection point for multiple lines. To minimize the size of interpolation errors, it is beneficial to chose the sphere radius as small as possible. Independent of those choices, data points generally tend to lie inhomogeneously distributed on the spheres. This reduces the interpolation accuracy of the uniformly parameterized spline surfaces.
Another question is how the uncertainty of lines can be expressed. When using starting points and direction vectors, a good way would be to place the starting point at the point of highest accuracy and store the corresponding (3×3) covariance matrix. In combination with the (3×3) covariance matrix of the direction vector, an appropriate representation of an uncertain line would be found. But then the starting points would lie arbitrarily distributed in space, which again dramatically reduces the interpolation accuracy of the corresponding spline surface.
Instead of using starting point and direction, lines in 3D space can also be represented by Plücker coordinates (Section 2.1.3). It is redundant as a vector with six parameters is used to describe one line. But due to the homogeneous character of this representation, it is possible to choose the vector length freely. This allows to put the line representations on selected subspaces of 6D space. If chosen adequately, this leads to a spatially more compact distribution which improves the interpolation accuracy of the resulting spline surfaces. When using homogeneous Plücker coordinates, subsequent calculations like triangulation (Section 2.4.4.1) or intersection with a plane (2.1.10) can be executed directly without the introduction of any additional parameters. Furthermore, uncertainties are conveniently described by a (6×6) covariance matrix without any need to transform the coordinates to a different form.

For these reasons, Plücker coordinates are chosen to represent lines in this work. A spline surface in 6D space through line data points will form the final continuous representation of the generic camera model. This representation is called *the surface model*.
The data points used to construct the spline surface will be euclideanly normalized (see Section 2.1.3). They therefore lie on a 5D subspace of 6D space. As Plücker coordinates have to fulfill the Plücker constraint (2.1.5) to be valid line representations, the dimension of the subspace is further reduced. Therefore, the data points lie on a 4D subspace of 6D space (see Figure 4.1.1 for an illustration). This actually matches the degrees of freedom of a line in 3D space.
The steps for creating a spline surface that represents a camera with known model parameters and mapping function are the following (also illustrated in Figure 4.1.1):

1. Choose a rectangular area in the camera image which is to be described by the model. The upper left corner lies at $(x_{min}, y_{min})^T$, marking the origin of the *spline coordinate system* SCS, and the lower right corner is at $(x_{max}, y_{max})^T$.

2. Select sample pixels $\boldsymbol{x}_s$ on an equidistant grid in the selected region of the image.

3. Determine the ray parameters $\mathbf{L}_s$ in Plücker coordinates for the selected pixels from the camera mapping.

4. Make sure the $\mathbf{L}_s$ fulfill the Plücker constraint and are euclideanly normalized. These are then the data points used to create the spline surface.

5. Use uniform parameterization to calculate the parameter values $(\bar{u}_k, \bar{v}_l)^T$ for each data point (see (2.3.20) for details). Specify the desired number of control points in u and v direction and determine the knot positions $\{u_0, ..., u_{n_k}\}$ and $\{v_0, ..., v_{m_k}\}$ (specified by (2.3.21)).

6. Determine the control points of the surface by spline approximation (Section 2.3.3).

These steps lead to the surface model as a continuous representation of the camera. It is based on data points taken from the camera mapping function. This function can be arbitrary, ranging from models of a simple linear pinhole camera to non-central catadioptric devices. How the surface model is used to achieve subpixel back projection and general forward projection is shown in the following sections.

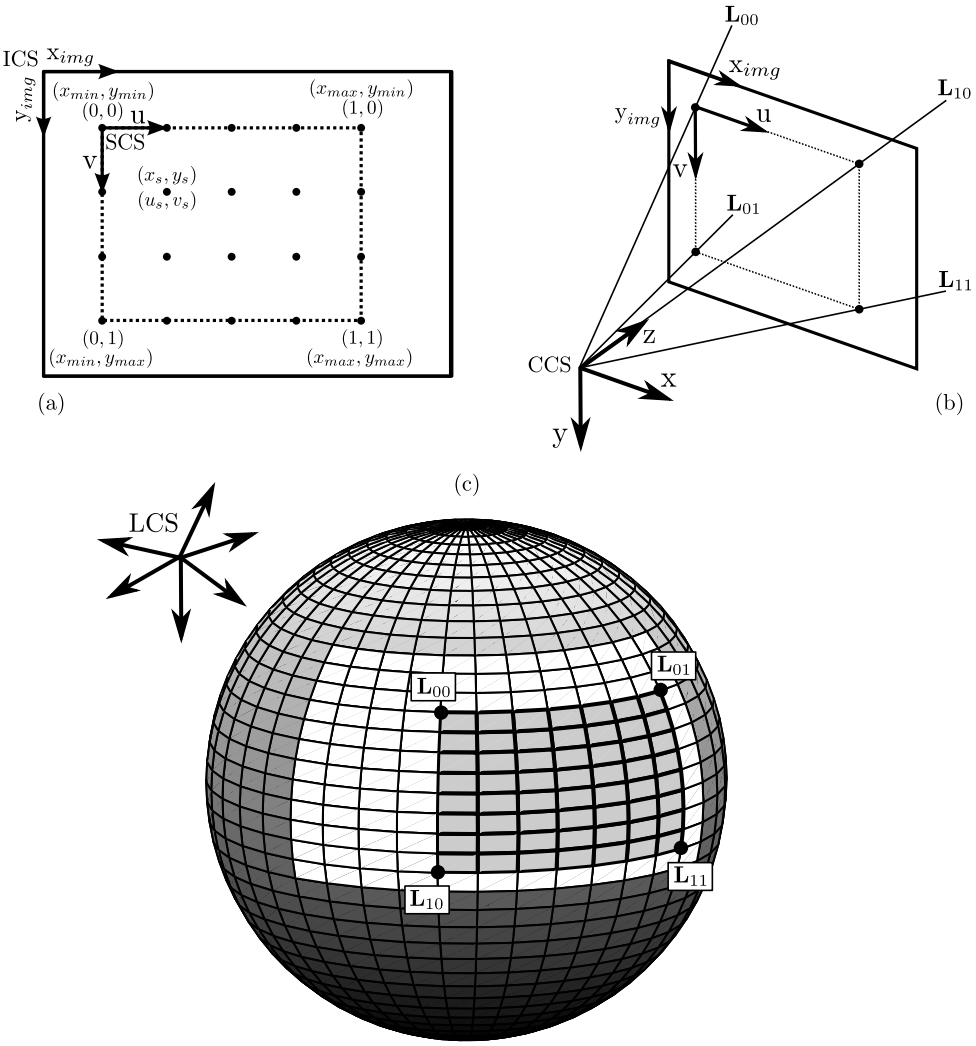


Figure 4.1.1.: Construction of the spline surface that serves as a camera representation. ICS, SCS, CCS and LCS are the image, spline, camera and line coordinate system, respectively.
(a) The camera image is sampled at equidistant positions $(x_s, y_s)^T$ within a rectangular area $(x_{min}, y_{min})^T \rightarrow (x_{max}, y_{max})^T$, which is the region of the image that is actually represented by the model.
(b) The corresponding line parameters $\mathbf{L}_s$ are determined, which serve as data points for the 6D spline surface (see text for details). In this illustration a pinhole model is used, the line coordinates $\mathbf{L}_{00}, \mathbf{L}_{10}, \mathbf{L}_{01}$ and $\mathbf{L}_{11}$ are shown as examples here.
(c) In 6D space, euclideanly normalized Plücker coordinates lie on a 5D subspace, which is visualized as a shaded sphere here. They fulfill the Plücker constraint, which puts them on a subspace of this sphere (white). Those lines that actually represent the camera are in yet another subspace, depicted in gray with thick lines.

4.2. Subpixel back projection with the surface model

Subpixel back projection is the procedure which determines the viewing ray for an arbitrary rational pixel position $(x_a, y_a)^T$ in the camera image. When the continuous spline surface model described in the last section is used, the corresponding parameter values $(u_a, v_a)^T$ have to be calculated first. Of course the pixel position has to lie within the modeled area, such as $x_{min} \leq x_a \leq x_{max}$ and $y_{min} \leq y_a \leq y_{max}$. As uniform parameterization is used during the construction of the spline surface, the relation between pixel positions and spline parameters is linear. This allows to determine the parameters via

$$u_a = \frac{x_a - x_{min}}{x_{max} - x_{min}} \tag{4.2.1}$$

and

$$v_a = \frac{y_a - y_{min}}{y_{max} - y_{min}}. \tag{4.2.2}$$

These serve to obtain the surface vector for the specified pixel coordinates from the spline by calculating

$$\mathbf{L}'_a = \mathbf{S}(u_a, v_a). \tag{4.2.3}$$

The spline surface lives in general unconstrainted 6D space. Therefore, surface points $\mathbf{L}'_a$ at arbitrary positions do generally not fulfill the Plücker constraint (2.1.5) and consequently are no valid line representations. Executing the procedure for orthogonalization depicted in Section 2.1.3 remedies this and delivers the final Plücker line coordinates $\mathbf{L}_a$.
This completes the procedure of subpixel back projection. Subsequent normalization, either spherically or euclideanly, is optional and will only be executed if necessary for the next task. The following section will analyze this procedure in terms of accuracy for different camera models.

4.2.1. Surface model accuracy evaluation

To test the applicability of the proposed concepts, a surface model is created for various simulated cameras. Sample rays are generated to build the 6D spline surface and back projected rays from arbitrary positions are then compared to the simulated ground truth rays to get a measure for the achieved accuracy. There are several aspects and multiple parameters which have an influence on the final shape of the spline surface. First of all, there is the choice of spline degree and the type of parameterization. Both will be fixed in this work. The degree is set to three, resulting in cubic basis functions. And as mentioned before, the parameterization will be uniform, which provides a linear relation between pixel position and spline parameters and therefore allows to easily transform one into the other (see (4.2.1) and (4.2.2)). The remaining parameters to choose are the distance d_d

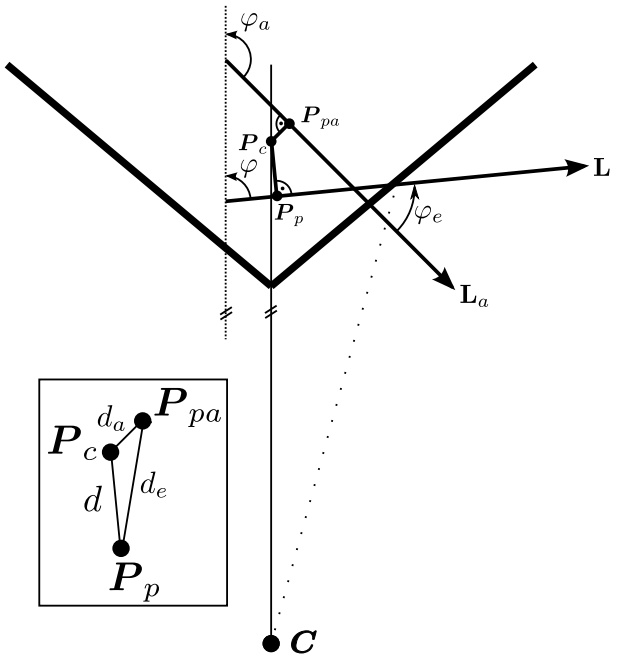


Figure 4.2.1.: The error values used for evaluating the model accuracy (in this illustration for a conic catadoptric camera). Shown are the simulated ground truth viewing ray $\mathbf{L}$ and the corresponding line $\mathbf{L}_a$ from the surface model. Line distance is represented by two values, namely the angular error φ_e and the position distance d_e. The latter is determined within the triangle of the points $\boldsymbol{P}_c$, $\boldsymbol{P}_p$ and $\boldsymbol{P}_{pa}$ which is shown in the magnified sketch in the bottom left (confer text for further explanations).

of the data point positions in the camera image and the number of spline control points $n+1$. The experiments will be conducted by executing the following scheme:

1. Select a camera type to be analyzed and choose realistic intrinsic parameters.

2. Create the surface model as described in the last section. The data points $\mathbf{L}_s$ come directly from the simulated camera. For evaluation purposes, the sample pixel positions $\boldsymbol{x}_s$ will be selected such as the size of the analyzed image area is maximized for the selected sample point position distance d_d. Furthermore, the area will be a square and the number of control points $n+1$ will always be equal in u and v direction. This simplifies the evaluation of the results.

3. Compare the back projected viewing rays from the surface model with the ground truth values for all pixels .

parameter	f	s	p_x	p_y	${}^c\mathrm{T}_v$
value	$6mm$	$5\mu m$	512	384	I_4

Table 4.2.1.: Intrinsic parameters of the simulated pinhole camera used to evaluate the accuracy of the surface model

All values used during evaluation are visualized in Figure 4.2.1. It shows the ground truth camera ray $\mathbf{L}$ and the line $\mathbf{L}_a$ from the surface model. Both are 6D vectors, but to simplify the evaluation, they will be compared by using two derived values: line angle and line position. The main criteria for selecting appropriate parameters for surface creation will be the angle φ_e between the two rays with

$$\varphi_e = \arccos\left(\frac{\boldsymbol{d}_a^T}{|\boldsymbol{d}_a|}\frac{\boldsymbol{d}}{|\boldsymbol{d}|}\right). \tag{4.2.4}$$

To get an impression of the spatial distribution of the ray directions, the angles φ and φ_a between the rays and the principal axis of the camera are calculated as well.
For non-central systems, also the ray distance d_e will be considered. It is determined with the help of the caustic center $\boldsymbol{P}_c$ which is the point that has minimum distance to all viewing rays. The distance between lines $\mathbf{L}$ and $\mathbf{L}_a$ is then defined via $d_e = |\boldsymbol{P}_p - \boldsymbol{P}_{pa}|$ with $\boldsymbol{P}_p$ and $\boldsymbol{P}_{pa}$ being the projections of $\boldsymbol{P}_c$ to both lines. The values φ_e and d_e are good measures for the evaluation of the model accuracy. However, for a better understanding of the results, occasionally also the differences of the single elements of $\mathbf{L}$ and $\mathbf{L}_a$ along specified lines in the camera image will be regarded.

4.2.1.1. Evaluation of a pinhole camera

The first camera to be modeled is a perfect pinhole camera without any lens distortions (Section 3.1) and a resolution of 1024×768 pixels. Forward projection is described by (3.1.10). For the experiments, camera parameters are chosen as shown in Table 4.2.1.
To get a general idea about the model accuracy, at first a data point position distance of 84 pixels is arbitrarily chosen. The considered image area is bounded by pixel coordinates $(134, 6)^T$ as upper left and $(890, 762)^T$ as lower right corner. The number of spline control points is set to the minimum of 4 in each direction. Figure 4.2.2 illustrates that the angle φ between the viewing rays and the principal axis increases with the distance from the principal point at the image center. Although a marginal number of control points are used, the corresponding values φ_a from the spline surface are very similar, as can be seen in Figure 4.2.3. Indeed, the model errors $|\varphi_e|$ are very small, with an average of $0.0032°$, a standard deviation of $0.0010°$ and a maximum of $0.0050°$. The spatial distribution of the angular error for the considered region can be seen in Figure 4.2.4. A closer look at the scanline marked in Figure 4.2.4 reveals the lateral distribution of the angular error

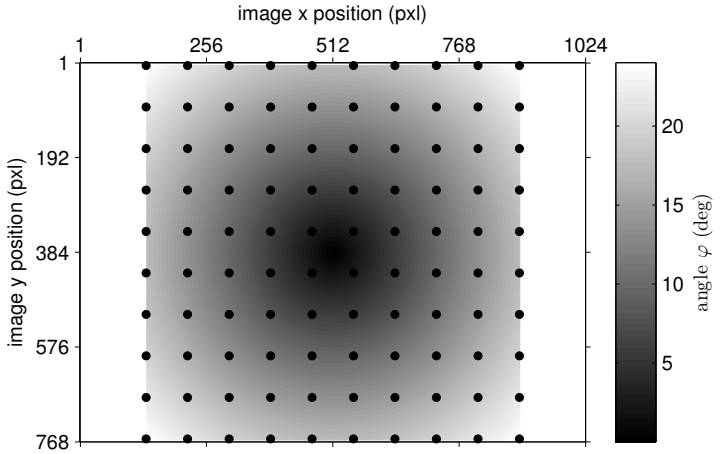


Figure 4.2.2.: Pinhole camera: simulated values of ground truth angles φ between the principal axis of the camera and the viewing ray for each pixel position. The positions of the data points used for spline surface construction are marked by black circles.

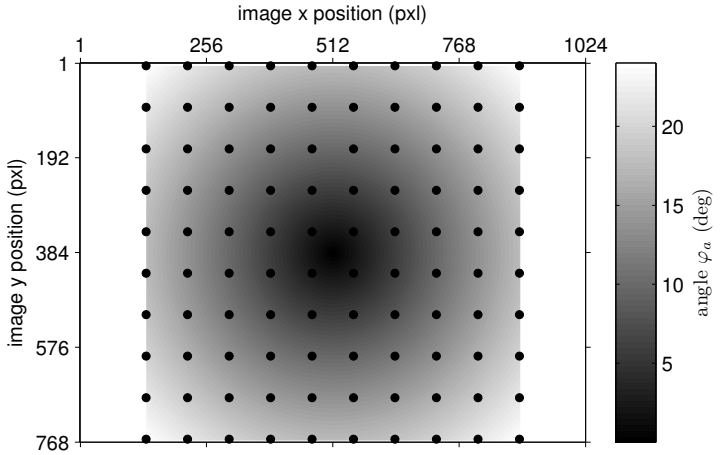


Figure 4.2.3.: Surface model of a pinhole camera: values of angles φ_a between the principal axis of the camera and the viewing ray for each pixel position. Data points were taken at positions marked with black circles. They have a distance of 84 pixels. The surface was created with only 4 control points in each direction. Differences to the ground truth from Figure 4.2.2 are marginal. They are visualized in Figure 4.2.4.

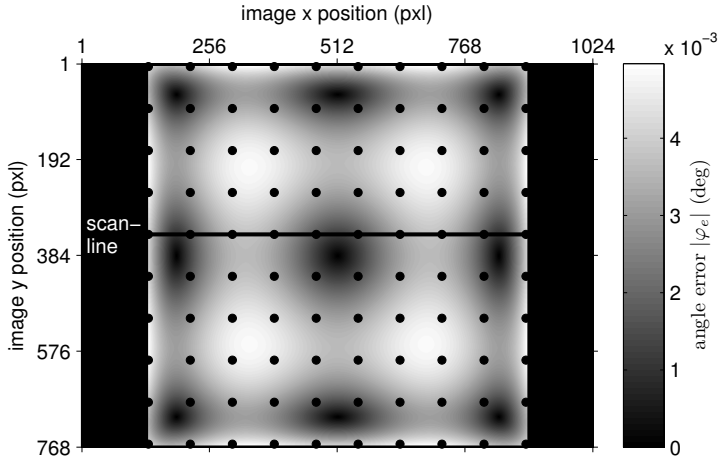


Figure 4.2.4.: Pinhole camera: difference $|\varphi_e|$ between the simulated ground truth ray angles and those from the surface model. The lateral profile along the marked scanline is shown in Figure 4.2.5

$|\varphi_e|$. It can be seen in Figure 4.2.5 and shows that the error varies significantly along the scanline. Although these results give a good impression of the accuracy of the continuous description they are generally biased by those steps which compute the angle φ_a from the actual spline surface points $\mathbf{L}'_a = \mathbf{S}(u_a, v_a)$. These effects are minimal in the case of the linear pinhole camera, nevertheless, the corresponding graphs are shown here to introduce the notations. In Figure 4.2.6, the comparison of the ground truth and the surface values can be seen for each dimension separately. There, the surface values $\mathbf{L}'_a$ are shown directly, without having applied any further processing steps to them. As all lines pass through the origin, the moment $\boldsymbol{m}$ is always the zero vector. Therefore, $\boldsymbol{m}'_a$ is free of any errors and L_4, L_5 and L_6 are not shown. Furthermore, orthogonalization is not necessary in this case ($\boldsymbol{d}'_a \times \mathbf{0}_3$ always equals $\mathbf{0}$) and no normalization was applied. The curves in Figure 4.2.7 reveal the error distributions of the single elements. They all finally contribute to the angular errors depicted in Figure 4.2.5.

The results shown so far were obtained with model parameters d_d and n fixed. If the goal is to achieve a specified accuracy, a threshold for the angular errors can be set and the parameters varied until the desired accuracy is reached. As the quality of the model for the linear pinhole camera is very high, it is possible to demand a relative tolerance of 0.001%. This is equivalent to a maximal measurement error of 1mm at a distance of 100m and corresponds to a maximal angular error of $5.7 \cdot 10^{-4^\circ}$. It is defined that the requirement is met when the mean angular error + two standard deviations is smaller than this value. Figure 4.2.8 visualizes the results with varying parameters and also marks those combinations that lie below the specified threshold. As less control points always lead to smaller matrices to be inverted during control point determination, it is wise to choose the parameters such as $n + 1$ is as small as possible. Additionally, a bigger data point position distance decreases the amount of sample points to be determined and reduces

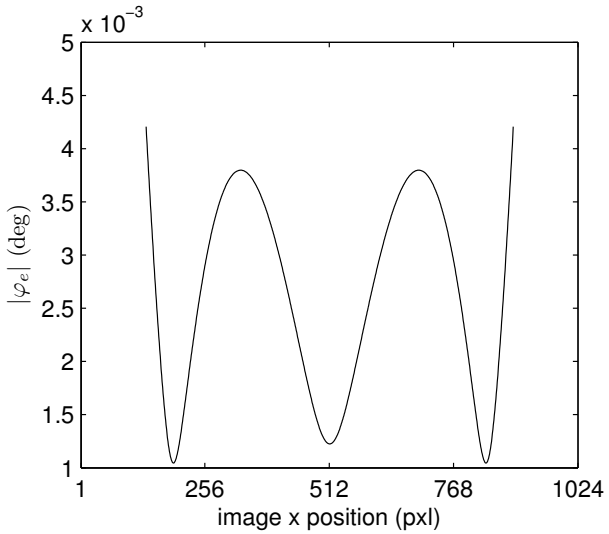


Figure 4.2.5.: Pinhole camera: profile of the angular error $|\varphi_e|$ along the image x position at $y = 342$.

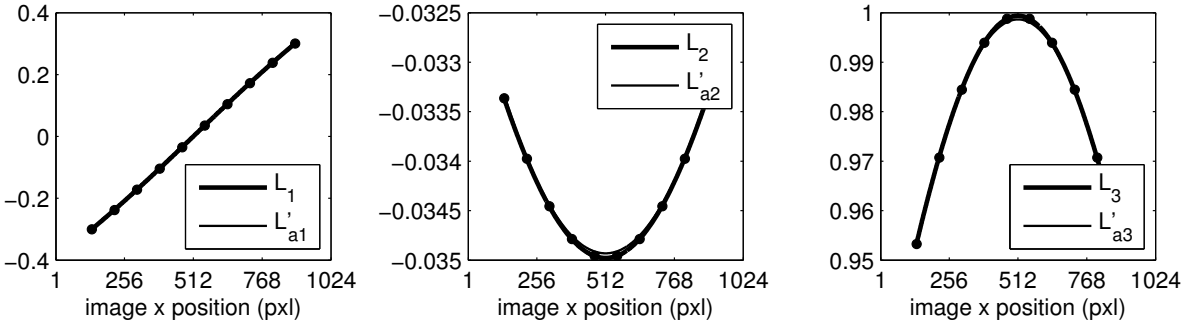


Figure 4.2.6.: Pinhole camera: profile lines along the image x position at $y = 342$ for the first three elements of $\mathbf{L}$ and $\mathbf{L}'_a$ separately. These are the direct results, without any post processing steps like e.g. normalization applied.

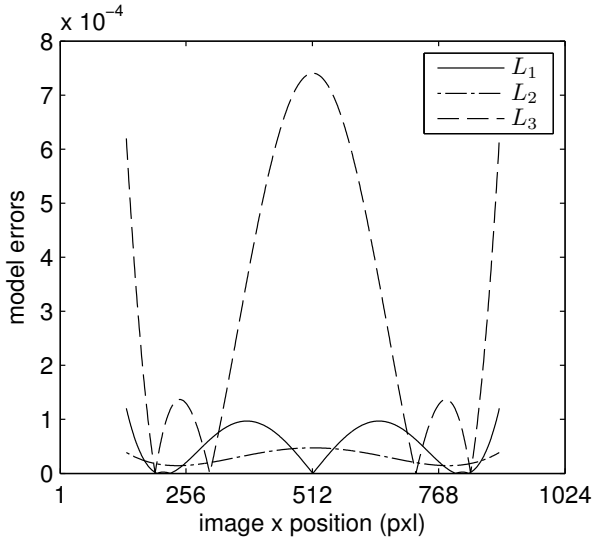


Figure 4.2.7.: Pinhole camera: first three elements of the model errors $|\mathbf{L}'_a - \mathbf{L}|$ along the profile line from Figure 4.2.4.

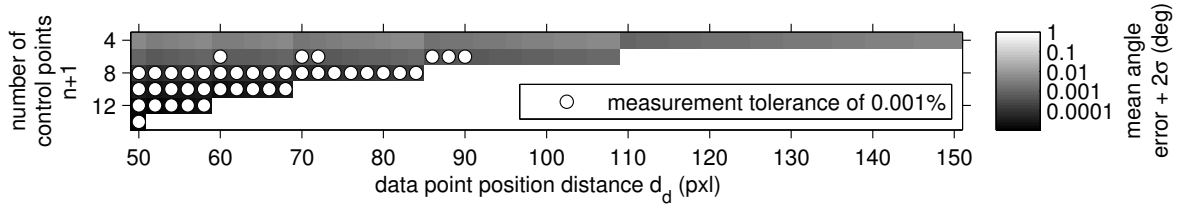


Figure 4.2.8.: Pinhole camera: the mean angular error + two standard deviations for different combinations of data point position distance d_d and spline control point numbers $n+1$. Combinations that have an angular error below 0.001% are marked with a circle.

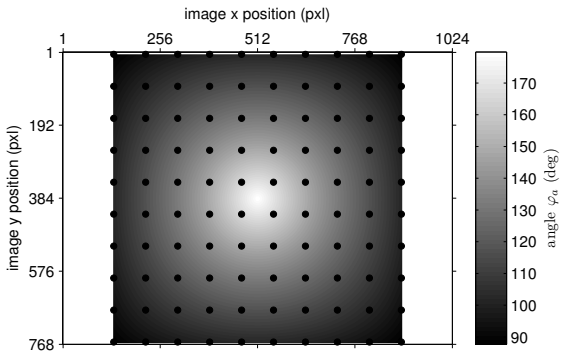
the number of rows in the equation systems to be solved. Therefore in this case $n+1 = 6$ and $d_d = 90$ would be a good choice for the parameter set.

4.2.1.2. Evaluation of a hypercatadioptric camera

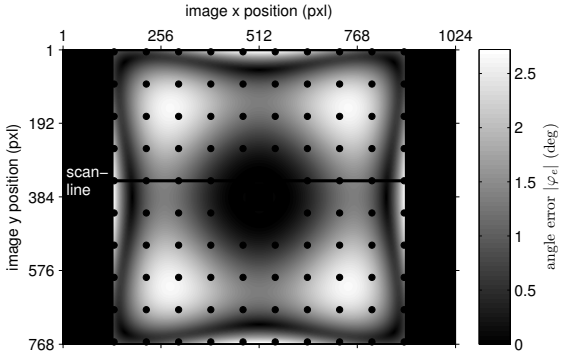
In this section, a catadioptric camera consisting of a perfect pinhole camera and a hyperboloid mirror (Section 3.2.2.2) will be analyzed. The parameters of the pinhole camera are the same as before (see Table 4.2.1). For the mirror, the parameters are set to $a = 12.5$mm and $\sqrt{a^2 + b^2} = 20$mm. To create a non-central setup, the camera is not placed in the second focal point of the mirror. Instead, it is positioned 10mm further away from the mirror. The effects on the viewing rays are qualitatively shown in Figure 3.2.8b.
Camera rays are determined at the same pixel positions as before, again 84 pixels apart. The constructed surface still has 4 control points in each parameter direction. Figure 4.2.9a shows the spatial distribution of φ_a for each pixel. Note that this time, due to the used mirror, the rays point into the opposite direction of the principal axis of the pinhole camera, leading to angles bigger than 90°. The model error φ_e is depicted in Figure 4.2.9b. A detailed lateral profile of the angular error can be found in Fig. 4.2.9c. It reveals that the greatest error on that scanline has a value of 2.356°. In fact, the overall average angular error is 1.2624° with a standard deviation of 0.7015° and a maximum value of 2.7209°, which are much bigger values than for the pinhole camera.
As the current setup is non-central, not only the angular, but also the positional accuracy of the viewing rays has to be regarded (Figure 4.2.1 illustrates how the different distances are determined). For the analyzed camera system, the caustic center $\boldsymbol{P}_c$ lies at $(0, 0, 51.28)^T$mm. The spatial distribution of the line distance d_a from $\boldsymbol{P}_c$ is shown in Figure 4.2.10a, the position error d_e in Figure 4.2.10b and Figure 4.2.10c depicts the error profile along the marked scanline. On average, the overall position error lies at 0.0261mm with a standard deviation of 0.0141mm and a maximum value of 0.0641mm.
Only the angular error is considered when choosing appropriate parameters d_d and n for the spline surface, because it has a much bigger influence on the accuracy of measurements taken by the camera. Figure 4.2.11 indicates that for achieving a measurement tolerance of 0.001% as before, as much as 12 control points with a data point position distance of 56 pixels would be needed. Lowering the requirement to a tolerance of 0.1%, corresponding to a measurement error of 10cm at a distance of 100m, would make 8 control points and a data point position distance of 72 pixels sufficient.

4.2.1.3. Conic camera

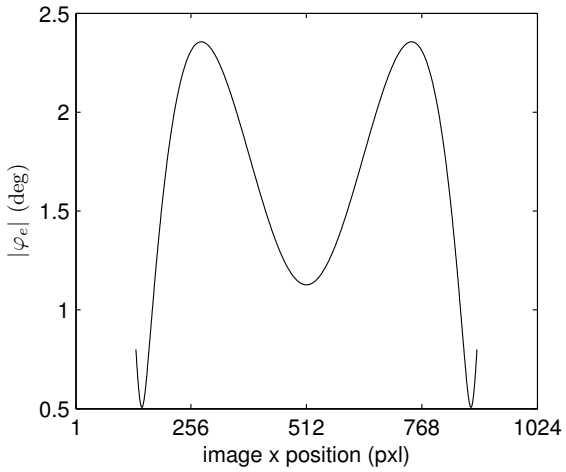
The same experiments as before are executed for a catadioptric camera with a conic mirror. Still, the utilized pinhole camera is configured as specified in Table 4.2.1. The mirror is



(a) Spatial distribution of the viewing ray angle φ_a from the surface model.

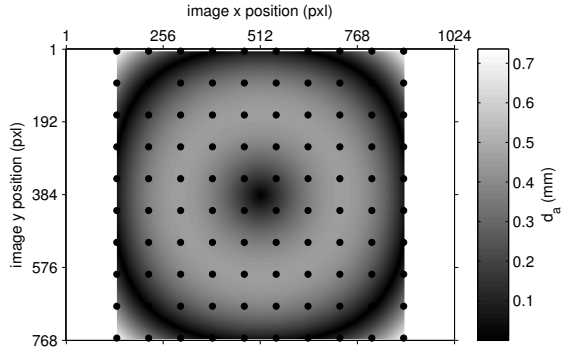


(b) Angular difference from ground truth.

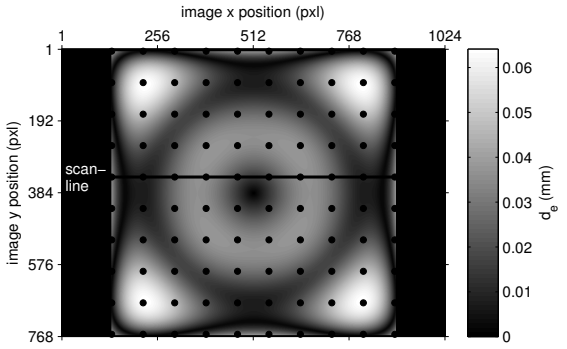


(c) Lateral profile of the angular error along the line at $y = 342$.

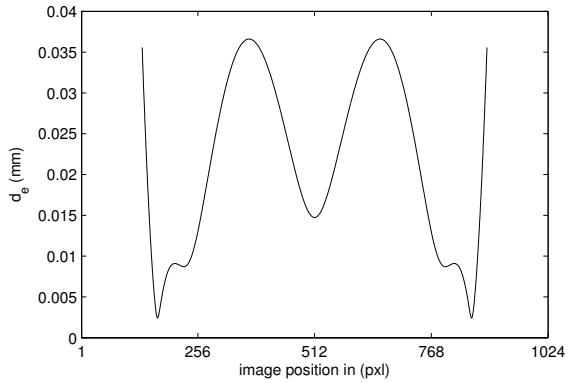
Figure 4.2.9.: Angular accuracy of the surface model for the non-central hypercatadioptric camera ($d_d = 84$, $n + 1 = 4$). Black circles in (a) and (b) mark the data point positions used for surface construction.



(a) Spatial distribution of the viewing ray distance d_a from the caustic center.



(b) Comparison with the ground truth rays gives the positional error d_e.



(c) Lateral profile of the positional error along the line at y=342.

Figure 4.2.10.: Positional accuracy of the surface model of a non-central hypercatadioptric camera ($d_d = 84$, $n + 1 = 4$). Black circles mark the data point positions used for surface construction.

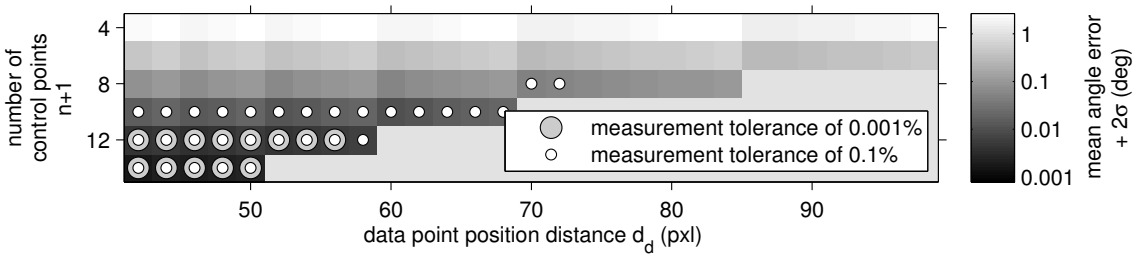


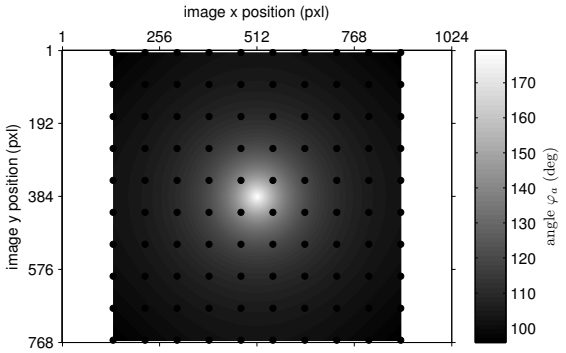
Figure 4.2.11.: Determination of spline surface parameters d_d and n for the non-central hypercatadioptric camera by identifying combinations where the depicted value lies below a specified threshold.

placed at a distance of 150mm along the principal axis and has an opening angle of $\alpha = 120°$. Figure 4.2.12a shows the distribution of the viewing ray angles, Figure 4.2.12b the angular error distribution and Figure 4.2.12c the lateral profile for the scanline at $y = 342$. The corresponding values for the ray position are illustrated in Figure 4.2.13. Here, a disadvantage of the method of spline surface creation becomes obvious. With the current approach of uniform parameterization, discontinuities can not be handled very well. As shown in Figure 4.2.14, the tip of the mirror creates a discontinuity in the camera mapping. This dramatically diminishes the accuracy of the surface model at the corresponding image position. The reason is twofold: on the one hand, more data and control points are required to follow quick changes. On the other hand, the distance between two data points that lie close to the discontinuity is comparably large. As stated before, the accuracy of a uniformly parameterized spline surface is best, when the data points are equidistant. These effects add up when keeping the small number of 4 spline control points with a data point position distance of 84 pixels as before. Then, the average angular error lies at 2.8734° with a standard deviation of 6.0171°.
Increasing the number of control points and at the same time using more data points from positions that lie closer together help constraining the negative effects to image areas in the vicinity of the discontinuity. This is shown in Figure 4.2.15. It can be seen that the error values decrease when more control points are used and that the ripples, which are an effect of the uniform parameterization, only appear closer to the image center. To achieve a maximal angular error of 1%, the number of control points has to be increased to 64, with the appropriate data point distance being 10 pixels (see Figure 4.2.16).

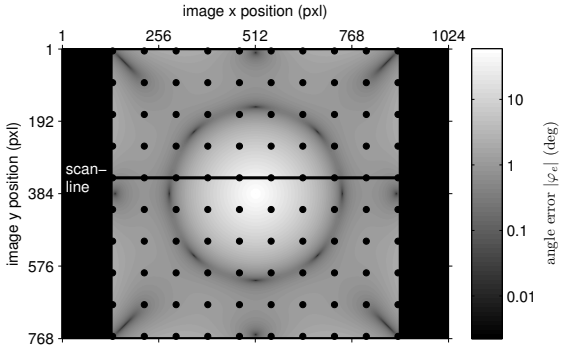
4.2.1.4. Realistic conic camera

The conic camera simulated in the last section can not exist in reality, because a) the area in the image center can not effectively be used as it is occluded by the pinhole camera and b) the used image region itself is likely to be circular, due to the restricted opening angle of the pinhole camera. This leads to a usable image area as depicted in Figure 4.2.17. Consequently, there are image pixels that do not have a viewing ray attached to them. The construction of a spline surface, however, requires data points on an equidistant rectangular grid. This leads to data point positions at which no real data is available. To avoid discontinuities, artificial data is generated for those positions by using procedures of interpolation and extrapolation. The procedures are arbitrarily chosen to be executed only along the x direction. For positions outside the viewing area, data is generated by determining the outmost known point $\mathbf{L}_1$ and the one next to it $\mathbf{L}_2$ and using them to build a linear extrapolation function. This allows to calculate new points at a distance x_d from the outmost point via

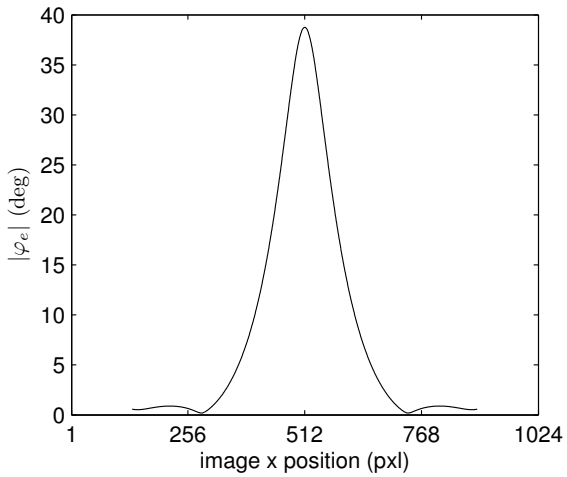
$$\mathbf{L}_e(x_d) = \mathbf{L}_1 + x_d(\mathbf{L}_1 - \mathbf{L}_2). \tag{4.2.5}$$



(a) Spatial distribution of the viewing ray angle.

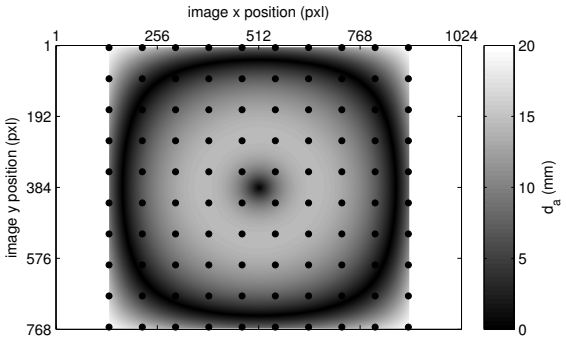


(b) The angular error.

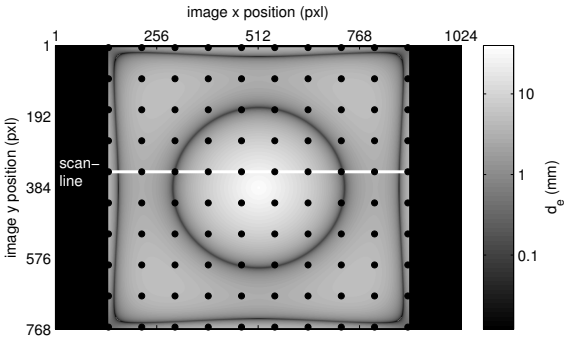


(c) Lateral profile of the angular error along the line at $y = 342$.

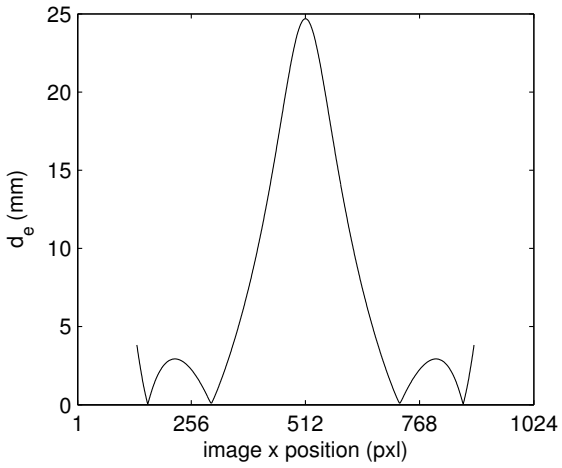
Figure 4.2.12.: Angular accuracy of the surface model of a conic camera ($d_d = 84$, $n + 1 = 4$). Black circles mark the data point positions used for surface construction.



(a) Spatial distribution of the ray position d_a (which is explained in Figure 4.2.1).



(b) The position error d_e.



(c) Lateral profile of the position error along the line at $y = 342$.

Figure 4.2.13.: Position accuracy of the surface model of a conic camera ($d_d = 84$, $n + 1 = 4$). Black circles mark the data point positions used for surface construction.

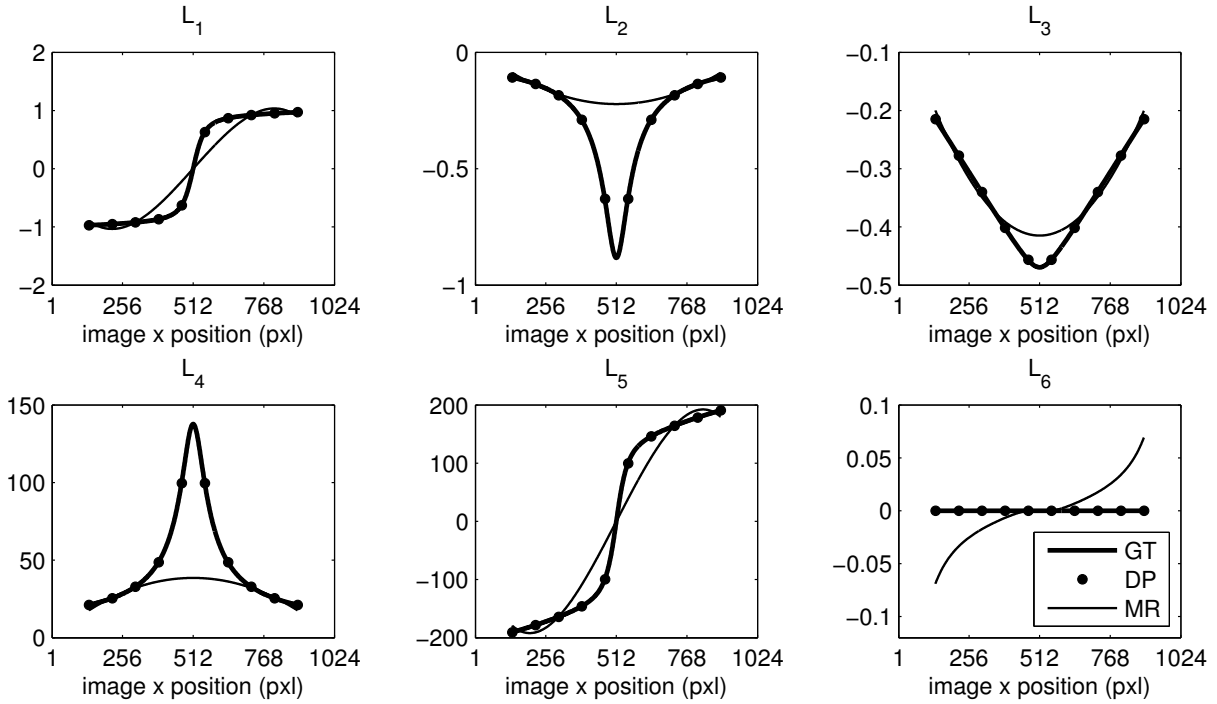


Figure 4.2.14.: Profiles of the line parameters $\mathbf{L}$ along the scanline at $y = 342$ for a conic camera (GT = ground truth, DP = data points, MR = model results). At the mirror tip, which lies at image position 512, discontinuities and quick changes of the camera mapping lead to big distances between the data points used to create the spline surface. It can be seen that a data point position distance of 84 pixels and the use of only 4 control points is not sufficient to create an adequate approximation of the mapping. L_6 differing from 0 for the MR is a consequence of the enforcement of the Plücker constraint.

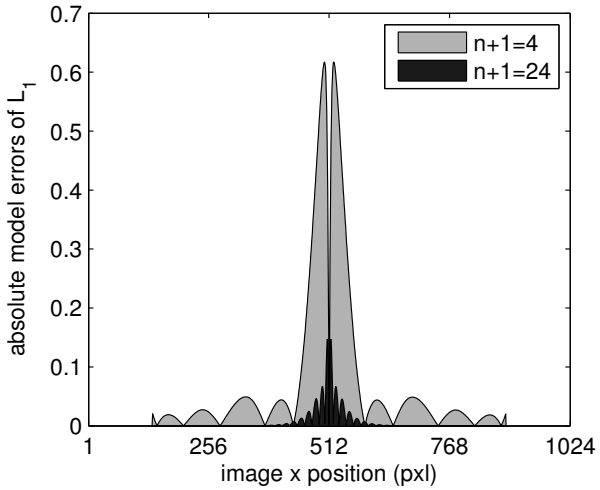


Figure 4.2.15.: Model error for the conic camera along the scanline at y=342. Increasing the number of control points reduces the absolute model errors and, even more importantly, constrains the negative effects to a smaller region around the discontinuity in the center.

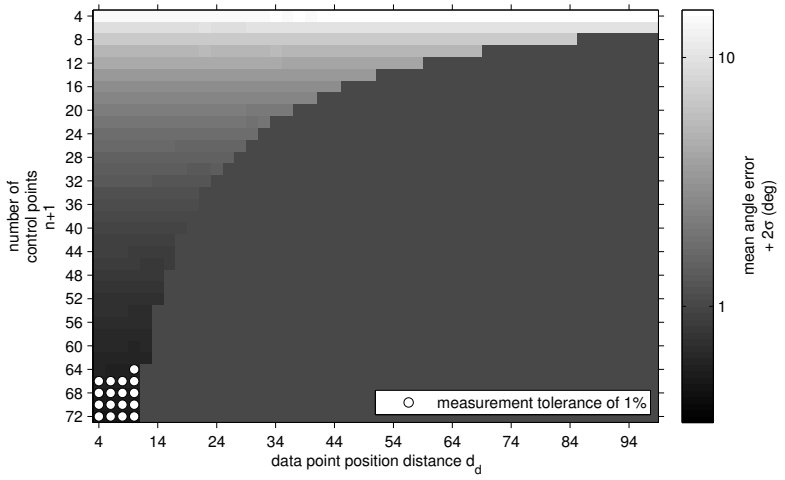


Figure 4.2.16.: Parameter selection for a conic camera. Due to the strong disturbances at the mirror tip, as much as 64 control points and a data point position distance of 10 pixels are needed to achieve a measurement tolerance of 1%.

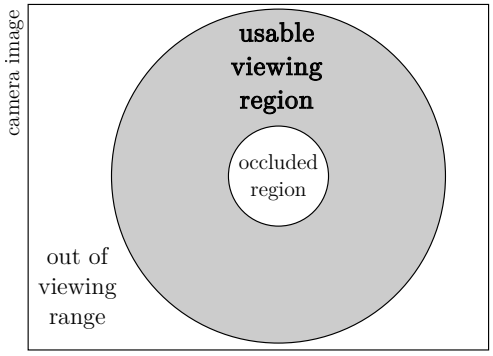
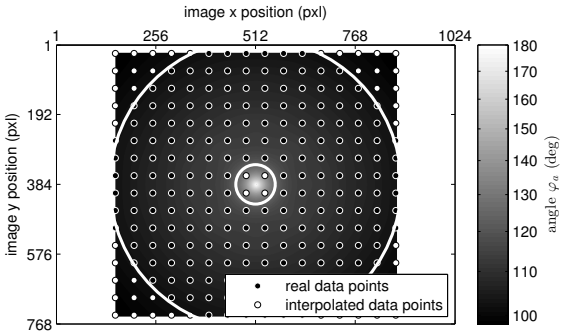


Figure 4.2.17.: The image region of a conic catadioptric camera that is actually used is bounded by the restricted opening angle of the pinhole camera on the outside and the area which is occluded by the camera in the center.

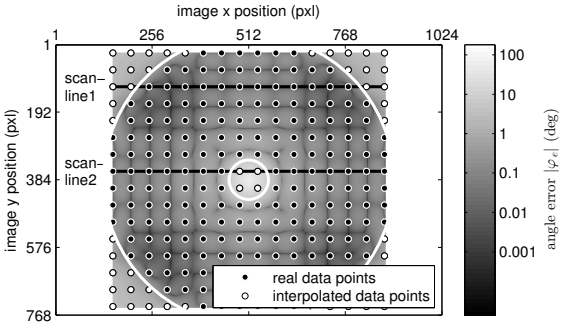
In the case of holes within the viewable region, missing data points are generated by linear interpolation. The points $\mathbf{L}_l$ on the left side of the hole and $\mathbf{L}_r$ on the right are used to generate points via

$$\mathbf{L}_i(x_d) = \mathbf{L}_l + \frac{x_d}{x_{lr}}(\mathbf{L}_r - \mathbf{L}_l). \tag{4.2.6}$$

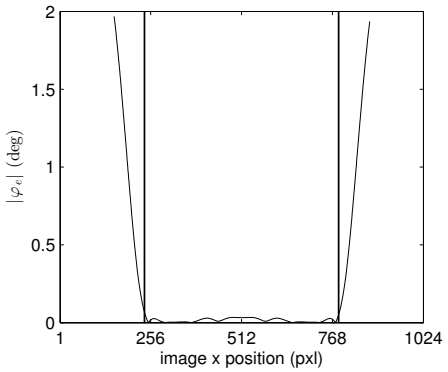
Here, x_{lr} is the data point position distance from $\mathbf{L}_l$ to $\mathbf{L}_r$ and x_d is the distance of the new data point position from $\mathbf{L}_l$. All extrapolated and interpolated data points $\mathbf{L}_e$ and $\mathbf{L}_i$ are taken directly as determined via the equations given above. They therefore do not necessarily fulfill the Plücker constraint and are no valid line representations. However, experiments showed that enforcing the Plücker constraint at this point severely reduces the accuracy of the final surface model. Also, these data points are not normalized in any way. The Plücker constraint will then be enforced during the back projection procedure to get valid line representations in the end. It is generally recommended to avoid getting ray parameters for those regions that were created with artificial data points. This advice is followed during evaluation of the realistic conic camera. Here, an occluded region with a radius of 40 pixels is defined in the image center and all points further than 380 pixels away from the central point are said to be out of the viewing range. This leads to an average angular error of 0.2684° with a standard deviation of 0.9353° for a data point position distance of 48 pixels and 14 spline control points. The corresponding spatial distributions can be found in Figure 4.2.18. The distribution of the ray position d_a, the positional error d_e and lateral profiles along two selected lines to illustrate the results of the extrapolation and interpolation procedures are shown in Figure 4.2.19.
To illustrate the result of the extrapolation and interpolation procedure, the profiles of the single elements of $\mathbf{L}_a$ along two different scanlines are shown in Figure 4.2.20 and Figure 4.2.21, respectively. There, the Plücker constraint is enforced, but no normalization is applied.
For the selection of appropriate parameters d_d and $n + 1$ for the creation of the spline surface, Figure 4.2.22 can be regarded. It shows that at least 16 control points and a data



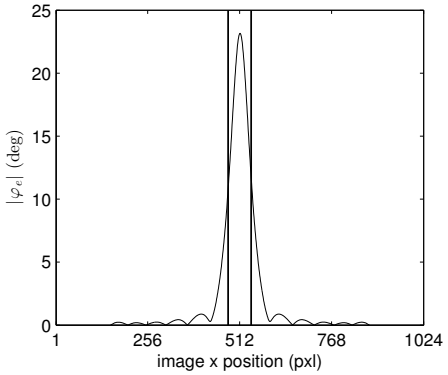
(a) Spatial distribution of the viewing ray angle.



(b) The angular model error.

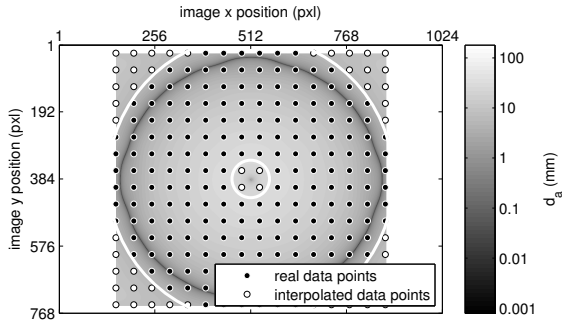


(c) Lateral profile of the angular error along scanline1 marked in (b). Data points outside the vertical lines were generated with linear extrapolation.

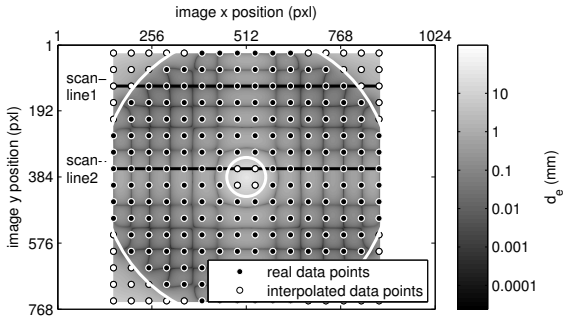


(d) Lateral profile of the angular error along scanline2 marked in (b). Data points between the vertical lines were generated via interpolation.

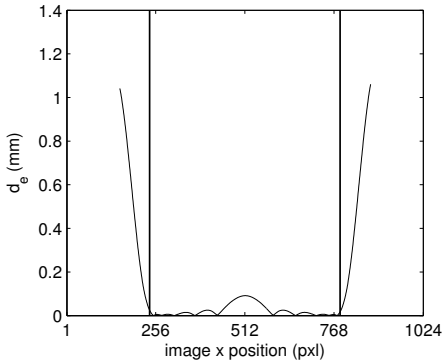
Figure 4.2.18.: Angular accuracy of the surface model with $d_d = 48$ pixels and $n + 1 = 14$ of a realistic conic camera. Positions with artificial (i.e. interpolated or extrapolated) data points are marked with a white-centered circle. Black-centered circles mark the data point positions where rays were taken from the camera simulation.



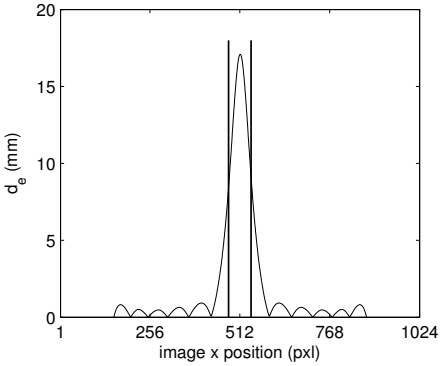
(a) Spatial distribution of the viewing ray position d_a (see Figure 4.2.1 for an explanation).



(b) The position error d_e when comparing the results of simulation and surface model.



(c) Lateral profile of the position error along scanline1 marked in (b). Data points outside the vertical lines were generated with linear extrapolation.



(d) Lateral profile of the position error along scanline2 marked in (b). Data points between the vertical lines were generated via interpolation.

Figure 4.2.19.: Position accuracy of the surface model with $d_d = 48$ pixels and $n+1 = 14$ of a realistic conic camera. Positions with artificial (i.e. interpolated or extrapolated) data points are marked with a white-centered circle. Black-centered circles mark the data point positions where rays were taken from the camera simulation.

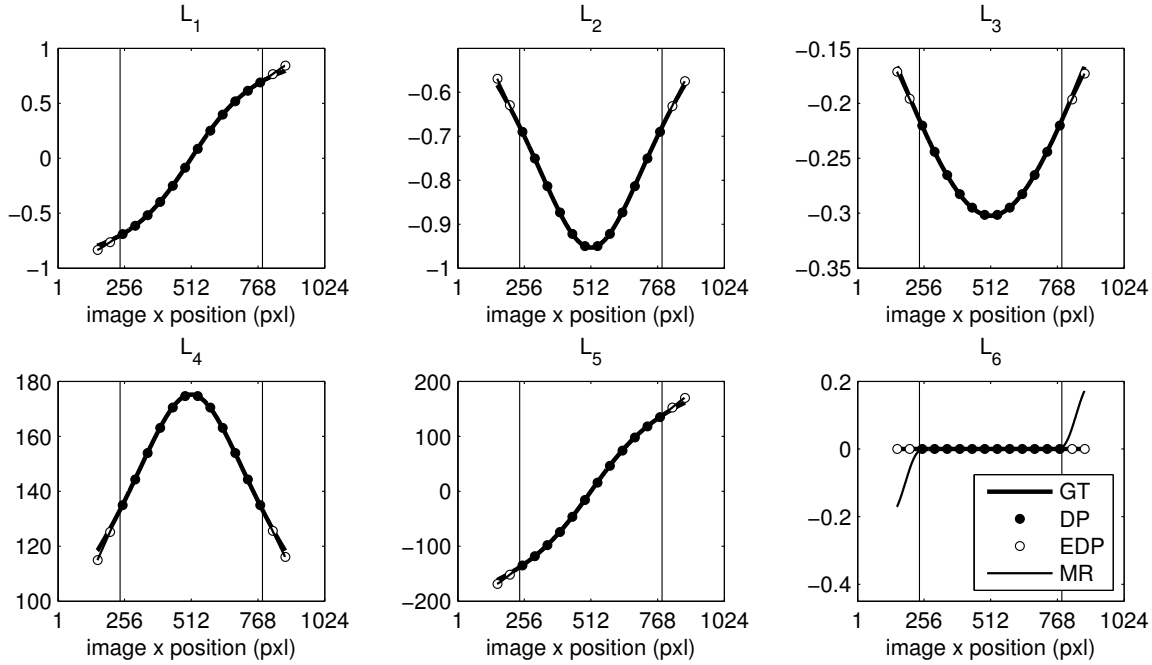


Figure 4.2.20.: Profiles of $\mathbf{L}_a$ along scanline1 from Figure 4.2.18b (GT = ground truth, DP = data poits, EDP = extrapolated data points, MR = model results).

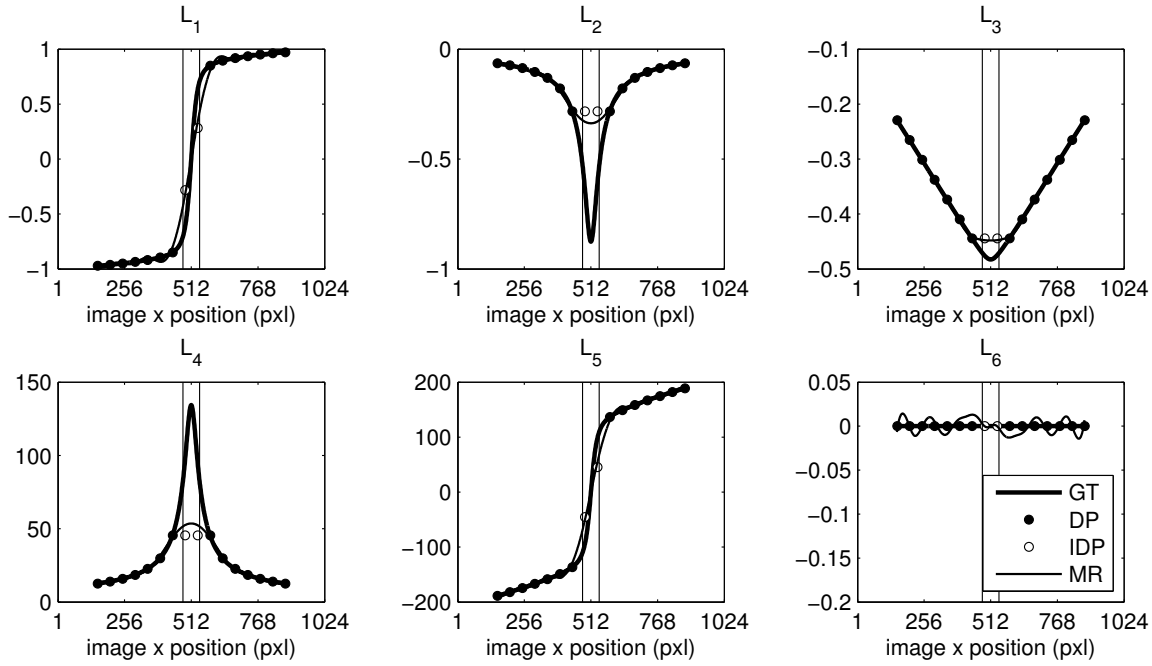


Figure 4.2.21.: Profiles of $\mathbf{L}_a$ along scanline2 from Figure 4.2.18b (GT = ground truth, DP = data poits, IDP = interpolated data points, MR = model results).

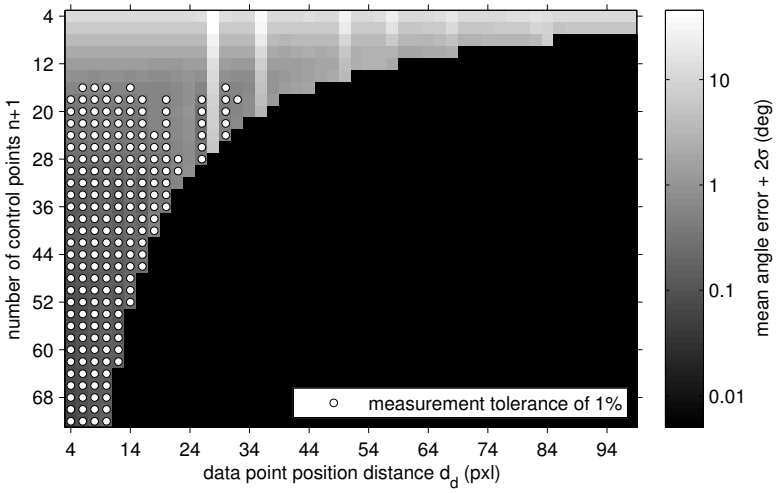


Figure 4.2.22.: Selection of appropriate spline parameters d_d and $n+1$ for the realistic conic camera. Error values from regions with interpolated and extrapolated data points are not included.

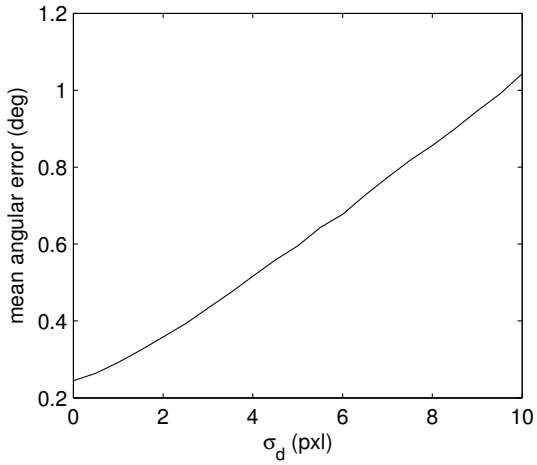
point position distance of 30 pixels is necessary to achieve a maximal angular error of 1%.

4.2.1.5. Conic camera with noisy data points

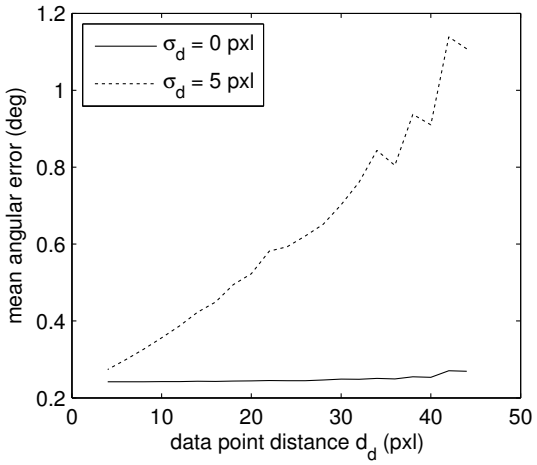
So far, all data points used for creating the surface model were free of any noise. Such perfect measurements can never exist in reality and therefore the effects of noisy data points on the model accuracy are analyzed here, exemplary for the conic camera. To simulate the effect of noise, the rays of slightly altered pixel positions $\boldsymbol{x}_{dn}$ are determined and assumed to be the noisy measurements for $\boldsymbol{x}_d$. Altering is done by adding an offset $\boldsymbol{x}_n$, which is a sample of a 2D Gaussian distribution, to the data point position:

$$\boldsymbol{x}_{dn} = \boldsymbol{x}_d + \boldsymbol{x}_n \quad \text{with} \quad \boldsymbol{x}_n \sim \mathsf{N}(\mathbf{0}_2, \Sigma_{dd}). \tag{4.2.7}$$

These conditions are realistic in the sense that usually a calibration procedure is executed before the model can be created and therefore all data points are subject to noise.
Not surprisingly, the noisy measurements have a negative influence on the model accuracy. When arbitrarily selecting a data point position distance of 24 and using 16 control points, the angular error increases with the noise. This can be seen in Figure 4.2.23a. The error values are obtained be determining the mean angular error 50 times for each noise level σ_d (the corresponding noise covariance matrices are defined as $\Sigma_{dd} = \mathrm{I}_2 \cdot \sigma_d^2$) and building the average.
Another useful result of these experiments is that under the presence of noise, more data



(a) Increasing noise leads to bigger angular errors, here shown for the conic camera and the model parameters $d_d = 24$ and $n + 1 = 16$.



(b) A decreasing data point position distance, i.e. an increasing number of data points, leads to a higher model accuracy. The number of control points is still fixed at 16.

Figure 4.2.23.: The influence of noise on the accuracy of the surface model for a conic camera.

points lead to better results for a fixed number of spline control points. For noise-free data points, their number has hardly any influence on the result, as can be verified in Figures 4.2.8, 4.2.11 and 4.2.16. A comparison of these two cases can be seen in Figure 4.2.23b. This leads to the conclusion that more data points will lead to a more accurate model, as the noise is then more likely to be canceled out by the spline approximation procedure.

4.2.2. Uncertain splines

So far, the surface model was based on data points without any uncertainty information. The last section showed, that noisy data influences the modeling accuracy. It would therefore be helpful to provide the user a model which also delivers uncertainty information. This defines the task to be solved in this section. From given data points with known uncertainty, a continuous description which delivers position and uncertainty information for intermediate positions is to be created. This description will be called the *uncertain surface model*.

As before, B-splines will be used to achieve this goal. Section 2.3.3 showed how a spline is fitted through given data points via approximation. Approximation means, that there are less control points than data points and that the spline does not necessarily pass through the data points. This is a desirable property as it leads to a smoother representation that

itself is less susceptible to measurement noise. As defined before, the control points matrix P is determined from the data point matrix Q via the equation system

$$\mathrm{A}_C\mathrm{P} = \mathrm{Q}. \tag{4.2.8}$$

Here, A_C contains the coefficients of the B-spline basis functions, P and Q are filled with the stacked control points $\boldsymbol{P}_i$ and data points $\boldsymbol{Q}_k$, respectively:

$$\mathrm{P} = \begin{bmatrix} \boldsymbol{P}_0^T \\ \boldsymbol{P}_1^T \\ \vdots \\ \boldsymbol{P}_n^T \end{bmatrix} \quad \mathrm{Q} = \begin{bmatrix} \boldsymbol{Q}_0^T \\ \boldsymbol{Q}_1^T \\ \vdots \\ \boldsymbol{Q}_{n_{qu}}^T \end{bmatrix}. \tag{4.2.9}$$

In the case of spline approximation, A_C is a non-square matrix. Assuming that all Q have the same accuracy, the least squares solution for P is therefore determined via

$$\mathrm{P} = (\mathrm{A}_C^T\mathrm{A}_C)^{-1}\mathrm{A}_C^T\mathrm{Q} \tag{4.2.10}$$

$$= \mathrm{A}_I\mathrm{Q} \quad \text{with} \quad \mathrm{A}_I = (\mathrm{A}_C^T\mathrm{A}_C)^{-1}\mathrm{A}_C^T. \tag{4.2.11}$$

To calculate the covariance matrix of P, it is convenient to reformulate the last equation. For a spline in d dimensions it can be rewritten as follows:

$$\begin{aligned} \mathrm{P} &= \mathrm{A}_I\mathrm{Q} \\ [\mathrm{P}_1\ \mathrm{P}_2\ \ldots\ \mathrm{P}_d] &= \mathrm{A}_I[\mathrm{Q}_1\ \mathrm{Q}_2\ \ldots\ \mathrm{Q}_d] \\ \Leftrightarrow \quad \begin{bmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ \vdots \\ \mathrm{P}_d \end{bmatrix} &= \begin{bmatrix} \mathrm{A}_I & 0 & \ldots & 0 \\ 0 & \mathrm{A}_I & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathrm{A}_I \end{bmatrix} \begin{bmatrix} \mathrm{Q}_1 \\ \mathrm{Q}_2 \\ \vdots \\ \mathrm{Q}_d \end{bmatrix}. \end{aligned} \tag{4.2.12}$$

The column matrices of control points and data points now explicitly read as

$$\begin{bmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ \vdots \\ \mathrm{P}_d \end{bmatrix} = \begin{bmatrix} p_{01} \\ p_{11} \\ \vdots \\ p_{n1} \\ p_{02} \\ \vdots \\ p_{n2} \\ p_{0d} \\ \vdots \\ p_{nd} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathrm{Q}_1 \\ \mathrm{Q}_2 \\ \vdots \\ \mathrm{Q}_d \end{bmatrix} = \begin{bmatrix} q_{01} \\ q_{11} \\ \vdots \\ q_{n_{qu}1} \\ q_{02} \\ \vdots \\ q_{n_{qu}2} \\ q_{0d} \\ \vdots \\ q_{n_{qu}d} \end{bmatrix}. \tag{4.2.13}$$

This leads to the global data point covariance matrix

$$\Sigma_{QQ} = \begin{bmatrix} \Sigma_{Q11} & \Sigma_{Q12} & \dots & \Sigma_{Q1d} \\ \Sigma_{Q21} & \Sigma_{Q22} & \dots & \Sigma_{Q2d} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{Qd1} & \Sigma_{Qd2} & \dots & \Sigma_{Qdd} \end{bmatrix}. \tag{4.2.14}$$

The Σ_{Qij} gather on their diagonal the elements from position (i, j) of the data point covariance matrices $\Sigma_{\boldsymbol{Q}_k\boldsymbol{Q}_k}$, $k \in [0, n_{qu}]$. For example with

$$\Sigma_{\boldsymbol{Q}_0\boldsymbol{Q}_0} = \begin{bmatrix} d_{0,11} & d_{0,12} \\ d_{0,21} & d_{0,22} \end{bmatrix} \quad \text{and} \quad \Sigma_{\boldsymbol{Q}_1\boldsymbol{Q}_1} = \begin{bmatrix} d_{1,11} & d_{1,12} \\ d_{1,21} & d_{1,22} \end{bmatrix} \tag{4.2.15}$$

this would lead to

$$\Sigma_{QQ} = \begin{bmatrix} d_{0,11} & 0 & d_{0,12} & 0 \\ 0 & d_{1,11} & 0 & d_{1,12} \\ d_{0,21} & 0 & d_{0,22} & 0 \\ 0 & d_{1,21} & 0 & d_{1,22} \end{bmatrix}. \tag{4.2.16}$$

The control point covariance matrix can now be determined by applying the standard linear approach for uncertainty propagation from (2.2.12) to the equation that determines the control points (4.2.11):

$$\Sigma_{PP} = \begin{bmatrix} \mathrm{A}_I & 0 & \dots & 0 \\ 0 & \mathrm{A}_I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{A}_I \end{bmatrix} \Sigma_{QQ} \begin{bmatrix} \mathrm{A}_I^T & 0 & \dots & 0 \\ 0 & \mathrm{A}_I^T & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{A}_I^T \end{bmatrix}. \tag{4.2.17}$$

Its contents are the covariances of all elements of the control point column matrix in (4.2.13). They are named as

$$\Sigma_{PP} = \begin{bmatrix} c_{00,11} & c_{01,11} & \dots & c_{0n,11} & c_{00,12} & & \\ c_{10,11} & c_{11,11} & \dots & c_{1n,11} & c_{10,12} & & \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \\ c_{n0,11} & c_{n1,11} & \dots & c_{nn,11} & c_{n0,12} & & \\ c_{00,21} & c_{01,21} & \dots & c_{0n,21} & c_{00,22} & & \\ & & & & & \ddots & \\ & & & & & & c_{nn,dd} \end{bmatrix} \tag{4.2.18}$$

which leads to the covariance matrix of a single control point $\boldsymbol{P}_i$

$$\Sigma_{\boldsymbol{P}_i\boldsymbol{P}_i} = \begin{bmatrix} c_{ii,11} & c_{ii,12} & \dots & c_{ii,1d} \\ c_{ii,21} & c_{ii,22} & \dots & c_{ii,2d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{ii,d1} & c_{ii,d2} & \dots & c_{ii,dd} \end{bmatrix}. \tag{4.2.19}$$

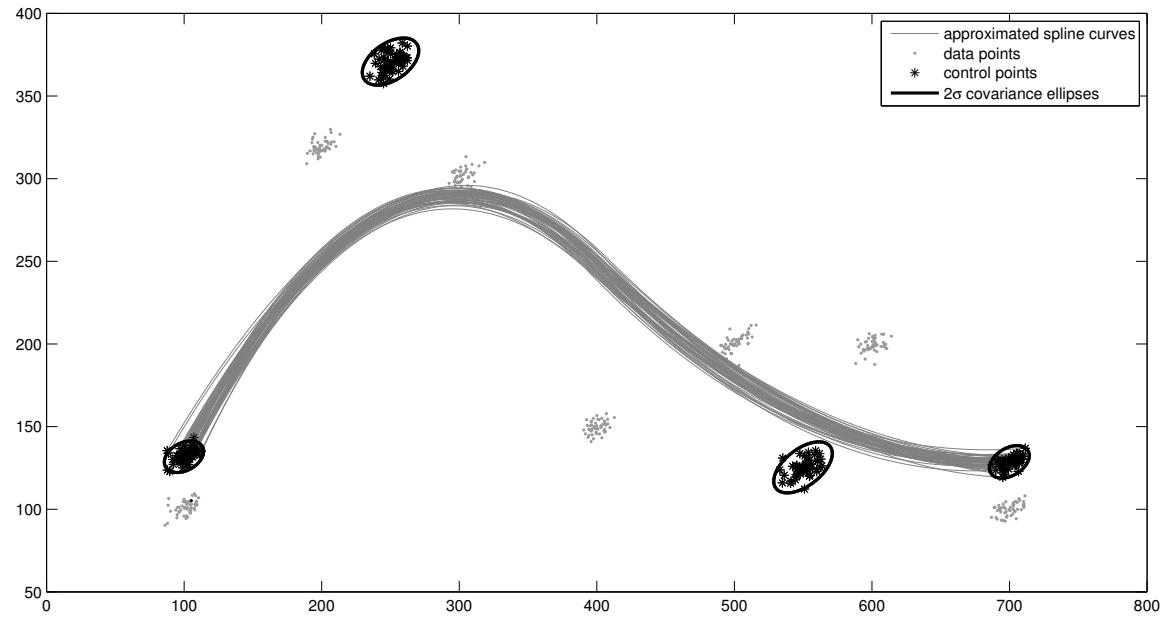


Figure 4.2.24.: Monte Carlo simulation of a spline curve approximation procedure with uncertain data points (gray). Only 50 of the 10.000 overall runs are shown. Uncertainties of the resulting control point clouds are visualized by the black 2σ ellipses. Calculating these covariance matrices of the control points via uncertainty propagation delivers results that are visually indistinguishable from the shown ellipses. This proves that linear uncertainty propagation is an adequate means to determine the curve uncertainty.

To verify these findings, Monte Carlo simulation is used (Section 2.2.2). The curve from Figure 2.3.1 is taken as an example. Each of the 7 data point $\boldsymbol{Q}_k$ is defined to be perturbed by Gaussian noise with a specified covariance matrix $\Sigma_{\boldsymbol{Q}_k\boldsymbol{Q}_k}$. For each set of samples, 4 control points $\boldsymbol{P}_i$ are determined via curve approximation. This procedure is repeated 10.000 times and the mean values $\boldsymbol{P}_i^{MC}$ and covariance matrices $\Sigma^{MC}_{\boldsymbol{P}_i\boldsymbol{P}_i}$ are calculated. A reduced selection of the results is shown in Figure 4.2.24. The uncertainties of the control points are visualized by the 2σ ellipses. Determining these uncertainties via uncertainty propagation reveals that the elements of the covariance matrices differ only by 2.21% on average. The propagated means have an average Mahalanobis distance of 0.0034 from the Monte Carlo results.
Having determined the covariance matrices of the control points, they can be used to get uncertainty information of any point $\boldsymbol{C}(u)$ of the spline curve. As each spline point is a linear combination of the control points, a derivation of the corresponding covariance matrix is directly possible. By identifying the $p+1$ control points with indices $\{s_u, ..., s_u + p\}$ that actually contribute to the point at parameter position u_c, the size of the involved

matrices can be dramatically reduced:

$$\boldsymbol{C}(u_c) = \sum_{i=0}^{n} N_{i,p}\boldsymbol{P}_i \tag{4.2.20}$$

$$= \sum_{i=s_u}^{s_u+p} N_{i,p}\boldsymbol{P}_i \tag{4.2.21}$$

$$= N_{s_u,p}\boldsymbol{P}_{s_u} + N_{s_u+1,p}\boldsymbol{P}_{s_u+1} + \ldots + N_{s_u+p,p}\boldsymbol{P}_{s_u+p} \tag{4.2.22}$$

$$= \underbrace{\begin{bmatrix} N_{s_u,p} & N_{s_u+1,p} & \ldots & N_{s_u+p,p} & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & N_{s_u,p} & \ldots & 0 \\ & & & & & \ddots & \\ & & & & & & N_{s_u+p,p} \end{bmatrix}}_{=\mathrm{N}_C} \begin{bmatrix} p_{su,1} \\ p_{su+1,1} \\ \vdots \\ p_{su+p,1} \\ p_{su,2} \\ \vdots \\ p_{su+p,d} \end{bmatrix}. \tag{4.2.23}$$

Via linear uncertainty propagation, the covariance matrix is now

$$\Sigma_{\boldsymbol{CC}} = \mathrm{N}_C \Sigma_{P_C P_C} \mathrm{N}_C^T \tag{4.2.24}$$

where $\Sigma_{P_C P_C}$ contains the covariances of all involved control points. Again, a Monte Carlo simulation is executed to verify this result, where 10.000 curves are fitted through perturbed data points. This time, the mean positions and covariance matrices of 101 points on the curve at parameter positions $u_i = \{0, 0.01, \ldots, 1.0\}$ are determined. These results are compared to the values obtained by fitting the curve through the mean data point positions and utilizing the corresponding covariances determined with (4.2.24). On average, the elements of the covariance matrices differ by 2.57% and the propagated means have a Mahalanobis distance of 0.0043 from the Monte Carlo results. This proves that uncertainty propagation serves to derive valid uncertainty information about curve points at arbitrary parameter positions.

The derivations to get the control point and surface point covariances for spline surfaces are similar to those for spline curves. They are not listed here because the notation becomes cumbersome and the results do not provide any further insights. Instead, an illustrative example for a spline surface in two-dimensional space is given. This will actually be used during the calibration procedure described in Chapter 5. There, uncertain chessboard corners are the data points $\boldsymbol{Q}_i$ that are used to create an uncertain spline surface which allows to get uncertain calibration plane positions for arbitrary pixels.

To assess the accuracy of the uncertainty propagation for spline surfaces, a Monte Carlo analysis equivalent to the one for spline curves is executed. Here, a spline surface is fitted through two-dimensional data points on a (7×7) grid which are again perturbed by varying Gaussian noise. The (5×5) control points are calculated by surface approximation. Figure 4.2.25 shows some exemplary data points, together with the resulting

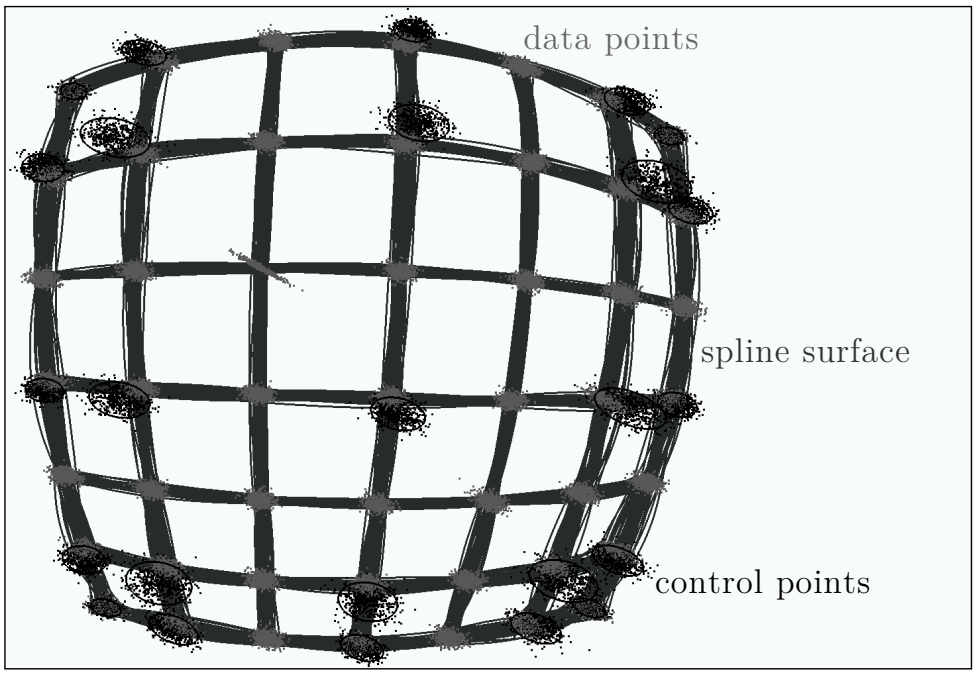


Figure 4.2.25.: Monte Carlo simulation of a spline surface approximation procedure with uncertain data points (only a fraction of all analyzed results is shown). Uncertainties of the control points are visualized by the 2σ ellipses.

control points and selected surface lines. The simulation with 10.000 iterations reveals an average error of 1.56% for the entries of the control point covariances and an average Mahalanobis distance of 0.0114 of the propagated means from the Monte Carlo means. For surface points at defined parameter positions on a (101×101) grid, the elements of the covariance matrices differ by 1.33% with a mean Mahalanobis distance of 0.0132. All findings are summarized in Table 4.2.2.
This section showed that the described procedure is an appropriate means for propagating uncertainties when approximating spline surfaces. It is therefore justified to use it to create an uncertain surface model as a representation of the generic camera model. The next section demonstrates this for selected simulated cameras.

4.2.3. Back projection with the uncertain surface model

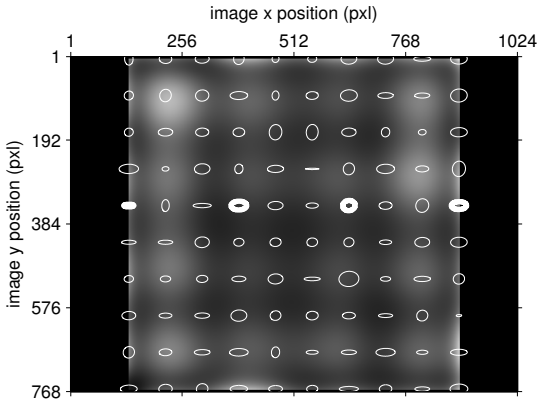
The method of creating an uncertain spline surface which was introduced in the last section will now be applied to camera modeling. As before, the surface model is a spline surface in 6D space, fitted through data points that are Plücker coordinates of lines from specified image pixel positions. Now, the aspect of uncertainty is added by providing a (6×6) covariance matrix for each data point. This allows to get line coordinates as well as

	spline curve		spline surface	
	covError	mahaDist	covError	mahaDist
control points	2.21%	0.0034	1.56%	0.0114
spline points	2.57%	0.0043	1.33%	0.0132

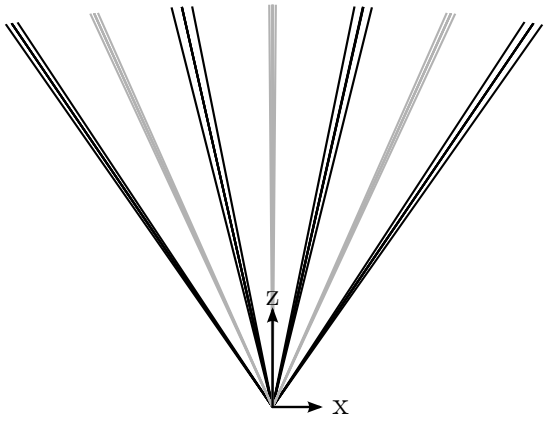
Table 4.2.2.: Accuracy analysis of the uncertainty propagation procedure for spline curves and surfaces. The values are explained in further detail in the text. covError = mean error of the entries of all covariance matrices when comparing Monte Carlo and propagated results. mahaDist = mean Mahalanobis distance of the propagated means from the Monte Carlo results.

the corresponding covariance matrix for arbitrary pixel coordinates. Therefore, the procedure of subpixel back projection is augmented by regarding the aspect of uncertainty. The applicability of this concept is demonstrated by building uncertain surface models of two cameras, namely a perfect pinhole camera and a realistic catadioptric camera with a conic mirror (see Section 4.2.1.4). In both cases, the data points are generated by simulation of line parameters for image pixel positions on an equidistant grid. To simulate uncertainty, 1000 further lines are obtained after adding samples of a 2D Gaussian distribution to each of the grid positions. The covariance matrices of these perturbed lines are then used to represent the line uncertainties. To get an impression of the varying uncertainty of the camera rays, the covariance matrix of every single pixel within the modeled region is determined and the corresponding traces are used to build a 2D profile.
The result for the pinhole camera which is created with a data point position distance of 84 pixels and a grid of (4×4) control points is shown in Figure 4.2.26a. There, the noisy data point positions are depicted by the 2σ ellipses. Figure 4.2.26b visualizes the corresponding viewing rays for those which are drawn with a thick white line. For illustrational purposes, the lines are projected to the xz-plane of the camera coordinate system. To show the uncertainty, samples are drawn from a Gaussian random variable that has the corresponding covariance matrix as its second moment. These samples lie on the 2σ hyperellipse of the 6D distribution. While the lines of the data points are shown in black, rays from intermediate positions, which are only available thanks to the uncertain surface model, are drawn with a gray color. This shows that determining ray parameters and their uncertainties for arbitrary pixel positions by using the concept of uncertain spline surfaces indeed gives valid results. The corresponding results for the realistic conic catadioptric camera from Section 4.2.1.4 can be seen in Figures 4.2.27a and 4.2.27b. For this example, a data point position distance of 48 pixels and a grid of (14×14) control points are used.

In this section, the concept of uncertain spline surfaces, which was introduced in Section 4.2.2, was applied to camera modeling. Uncertain surface models of two selected cameras were created and qualitative experiments showed that they can be used for generic camera modeling in terms of subpixel back projection. The next section will describe how general forward projection can be achieved and prove the applicability of that concept.

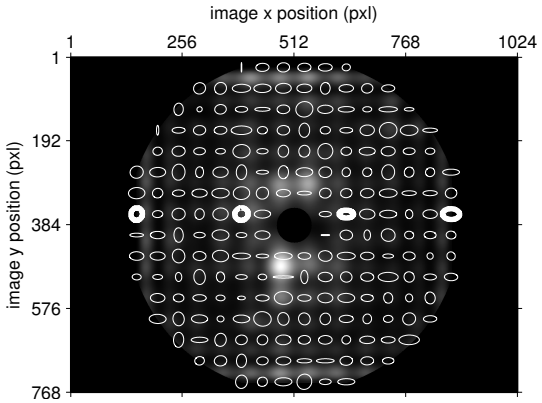


(a) The uncertainty of the viewing rays are represented by the traces of the (6×6) covariance matrices for each pixel. Bright colors stand for high uncertainties. Data point position uncertainties are represented by the 2σ ellipses.

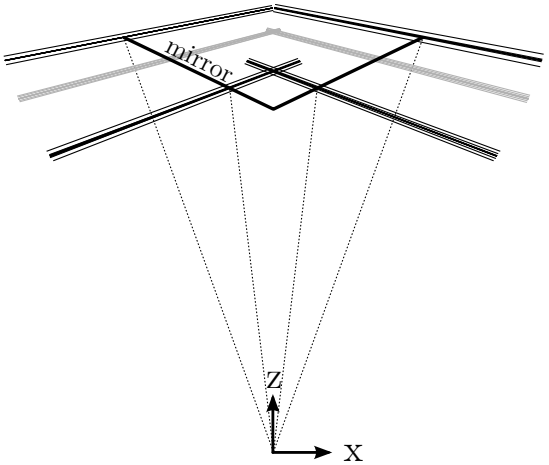


(b) Exemplary viewing rays and their uncertainties (black lines) for the data points marked by thick white ellipses in the left image. The uncertain surface model also allows to get uncertain lines for arbitrary intermediate positions (gray lines).

Figure 4.2.26.: The uncertain surface model of a pinhole camera: the uncertainties of the data points are taken into account. This allows to get ray parameters and covariance matrices for arbitrary image positions.



(a) Traces of the (6×6) covariance matrices for each pixel. Bright colors stand for high uncertainties. Data point position uncertainties are represented by the 2σ ellipses.



(b) Viewing rays and their uncertainties (black lines) for the data points marked by thick white ellipses in the left image, and for arbitrary intermediate positions (gray lines).

Figure 4.2.27.: Results of the uncertain surface model of a realistic conic camera.

4.3. General forward projection with the surface model

In the last section it was shown how the surface model can be used to determine a viewing ray for arbitrary pixel positions in the camera image. The procedure is called subpixel back projection and can help when realizing a method for general forward projection, which will be introduced in this section. General forward projection is the method that determines for an arbitrary 3D point $\boldsymbol{P}_{cam}$ the pixel position $\boldsymbol{p}_{img}$ which it is mapped to by the imaging system. $\boldsymbol{P}_{cam}$ is given in the camera coordinate system and has to lie in the viewing range of the camera. Traditional models based on the pinhole camera are commonly described in the forward direction, as they first project a point to the image plane, then apply a distortion function and shift and scale the result to get the final pixel position (see Section 3.1.2 for details). Other models, e.g. Scaramuzza's (Section 3.2.1.2), are defined as back projection functions. The mathematical effort of their inversion to achieve forward projection depends on the complexity of the model, e.g. the polynomial degree of the distortion function.
One property greatly simplifies the process for these models: their centrality. The existence of a projection center provides a point that lies on every viewing ray. This allows to determine all line parameters directly, which can then be used in the projection process. For non-central cameras, however, the direction of the viewing ray remains unknown. This complicates the projection procedure because valuable information is missing, which would be helpful to e.g. determine intersection points with calibration planes (two-plane model, Section 3.3.1). If two spline surfaces $\boldsymbol{S}_s$ and $\boldsymbol{S}_d$ are used to describe rays like proposed in [RW12b] the goal is to identify the correct parameter pair $(u_f, v_f)^T$ for which the corresponding ray $\boldsymbol{r}_f$ passes through the point $\boldsymbol{P}_{cam}$. A possible approach is then to minimize the distance $d_f(\boldsymbol{r}_f, \boldsymbol{P}_{cam})$ between ray and point. This leads to the optimization problem

$$(u_f, v_f)^T = \underset{u,v}{\operatorname{argmin}}\, d_f(\boldsymbol{r}_f(u, v), \boldsymbol{P}_{cam}) \tag{4.3.1}$$

which can be solved by using a non-linear optimization method like Levenberg-Marquardt. By inverting the equations which connect spline parameters with pixel positions (shown in (4.2.1) and (4.2.2)), the desired pixel coordinates $\boldsymbol{p}_{cam} = (x_f, y_f)^T$ can finally be determined via

$$x_f = u_f(x_{max} - x_{min}) + x_{min} \tag{4.3.2}$$

and

$$y_f = v_f(y_{max} - y_{min}) + y_{min}. \tag{4.3.3}$$

To gain valid initial values for the optimization procedure, rays $\boldsymbol{r}_s$ can be calculated at predefined parameter positions $(u_s, v_s)^T$ and the pair with the smallest corresponding distance d_f is chosen as starting position. The generic model allows a scene point to be seen at multiple positions in the camera image. Having this in mind, not only the best, but all rays $\boldsymbol{r}_s$ with $d_f(\boldsymbol{r}_s, \boldsymbol{P}_{cam})$ below a chosen threshold ϵ_d can be used to initialize the

optimization process. The results of those procedures that converge can than be considered as projections of $\boldsymbol{P}_{cam}$. Experiments conducted in [RW12b] reveal the high accuracy of this approach and confirm its applicability to the problem of general forward projection.

In this work, the camera is represented by the surface model, which is a spline surface in 6D Plücker space. A complete Plücker vector constructed from $\boldsymbol{P}_{cam}$ and an optical center would allow to directly invert the spline surface with the procedure described in Section 2.3.1. As this is not possible, another method has to be found. Therefore, the relation of the incidence of a line and a point will be used here. It states that a point $\mathbf{P}_{cam}$ lies on a line with Plücker coordinates $\mathbf{L}$ in case its product with the dual Plücker matrix $\overline{\Gamma}(\mathbf{L_f})$ (see (2.1.8) for the definition) becomes the zero vector:

$$\overline{\Gamma}(\mathbf{L_f})\mathbf{P}_{cam} = \mathbf{0}_4. \tag{4.3.4}$$

Here, the line parameters come from the spline surface model $\mathbf{S}(u, v)$. This is used to build the cost function

$$m(u, v) = \boldsymbol{m}^T(u, v)\boldsymbol{m}(u, v) \tag{4.3.5}$$

where

$$\boldsymbol{m}(u, v) = \overline{\Gamma}(\mathbf{S}(u, v))\mathbf{P}_{cam}. \tag{4.3.6}$$

Derived from this follows the minimization problem

$$(u_f, v_f)^T = \underset{u,v}{\operatorname{argmin}}\ \boldsymbol{m}^T(u, v)\boldsymbol{m}(u, v). \tag{4.3.7}$$

The parameter pair that solves this problem needs to be converted to pixel coordinates by (4.3.2) and (4.3.3) to get the desired image position $\boldsymbol{p}_{img}$. This completes the task of general forward projection with the surface model.
A remaining issue is the actual solution of (4.3.7). The problem is non-linear and made more complicated by the piecewise character of the spline surface. Any non-linear optimization numerical procedure would generally serve to determine the desired parameters (e.g. Levenberg-Marquardt). In this work, the numerical downhill simplex method of John Nelder and Roger Mead [NM65] is used. It has the advantage that no derivatives need to be determined and that it converges more reliably in case the optimization space exhibits long valleys with small gradients (which occasionally occurs in practice).
For most common imaging systems, forward projection is an injective process. All cameras considered in this work are of that kind (including the catadioptrics). Therefore, forward projection will always have a unique solution. Of course there are also systems where a scene point is projected to multiple pixels, e.g. multi-camera setups or catadioptrics with non-convex mirrors. In those cases, the geometry of the imaging system itself causes that (4.3.7) does not have a single global solution. The optimization procedure then has to take this possibility into account, e.g. as described before by starting it from multiple possible initial values.

Another reason for (4.3.7) having multiple solutions can be inaccuracies introduced by the spline surface, especially when they appear as oscillations. Noisy measurements used during the calibration procedure can have the same effect.
Mathematically, the elements of $\boldsymbol{m}^T\boldsymbol{m}$ are a multiplication of the spline surface basis functions and the entries of $\mathbf{P}_{cam}$. The basis functions are polynomials, in this work of third order, and their coefficients change at the knot parameter positions. Consequently, $\boldsymbol{m}^T\boldsymbol{m}$ is a sum of squared polynomials of third order, which are defined piecewise over the parameter space. Global convexity is therefore not guaranteed. For this reason, it is essential to identify a good starting point for the optimization procedure. The following approach is taken in this work: at defined sample positions $(u_{si}, v_{si})^T$ in the parameter space the first few iterations of the Nelder-Mead procedure are executed to get closer to a nearby minimum. Two sample points are merged in case their resulting positions in parameter space lie close together. For the remaining positions with the lowest cost value $\boldsymbol{m}^T\boldsymbol{m}$, the full Nelder-Mead optimization is executed. The parameter position belonging to the smallest cost value is accepted as the solution for the optimization problem. It is used to determine the final pixel position $\boldsymbol{p}_{cam} = (x_f, y_f)^T$ which is the result of the forward projection procedure.
To prove the applicability and the accuracy of the described approach, it is tested for the three different camera models already introduced before for the evaluation of the back projection procedure: the pinhole camera from Section 4.2.1.1, the hypercatadioptric camera from Section 4.2.1.2 and the conic camera from Section 4.2.1.3. In each case, a surface model is created with selected values for the data point position distance d_d and the number of control points $n + 1$. This can be used to determine values for the cost function $m(u, v)$ from (4.3.5) with arbitrary point positions $\boldsymbol{P}_{cam}$. Experiments showed, that the minimum of $m(u, v)$ can be difficult to identify due to small and long valleys in the parameter space. Omitting the last element of $\boldsymbol{m}(u, v)$ alleviates this problem. The reason is that only the values of the moment vector of the Plücker coordinates contribute to this element, which tend to be much bigger than those of the direction vector. Therefore, all experiments are conducted with the modified cost function

$$m' = \boldsymbol{m}_{1..3}^T\boldsymbol{m}_{1..3} \tag{4.3.8}$$

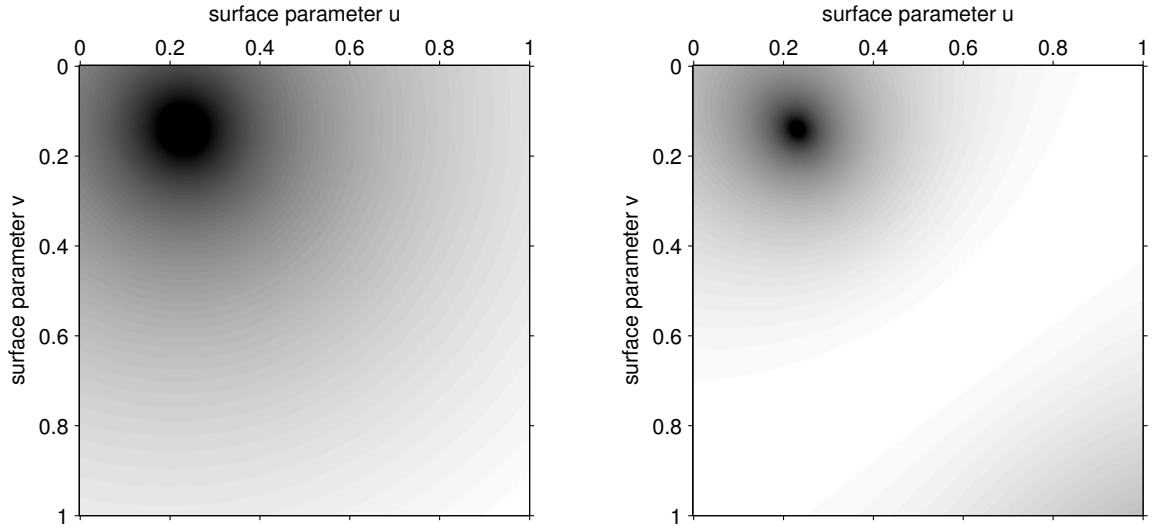
where $\boldsymbol{m}_{1..3}$ is a vector consisting of the first three elements of $\boldsymbol{m}$.
Exemplary points $\boldsymbol{P}_{cam_e}$ in camera coordinates which are needed to test the proposed method are generated by the following procedure:

1. A pixel position $\boldsymbol{p}_{img_e}$ is selected.

2. It is converted to the parameter position $(u_e, v_e)^T$ via (4.2.1) and (4.2.2).

3. The corresponding viewing ray parameters $\mathbf{L}_e = \mathbf{S}(u_e, v_e)$ are determined.

4. The point $\boldsymbol{P}_{cam_e}$ is defined as the one that lies on the viewing ray, 10m away from



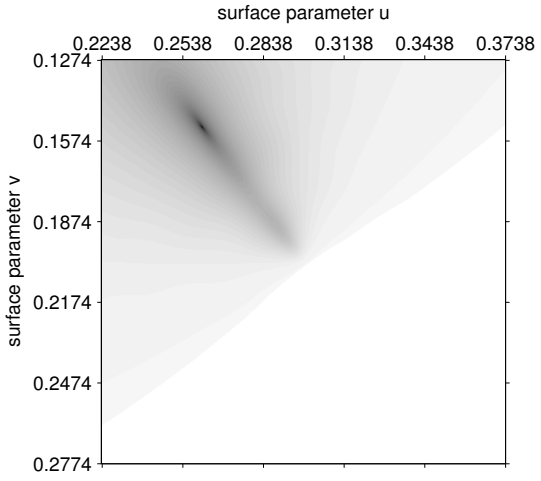
(a) The cost function $m'(u, v, \boldsymbol{P}_{cam_e})$ for the complete parameter space of a pinhole surface model. $\boldsymbol{P}_{cam_e}$ was generated from pixel position $\boldsymbol{p}_{img_e} = (256, 192)^T$. There is a single minimum exactly at $(u_e, v_e)^T$.

(b) $m'(u, v, \boldsymbol{P}_{cam_e})$ for a hypercatadioptric surface model. $\boldsymbol{P}_{cam_e}$ was generated from pixel position $\boldsymbol{p}_{img_e} = (256, 192)^T$. A second minimum appears in the lower right corner.

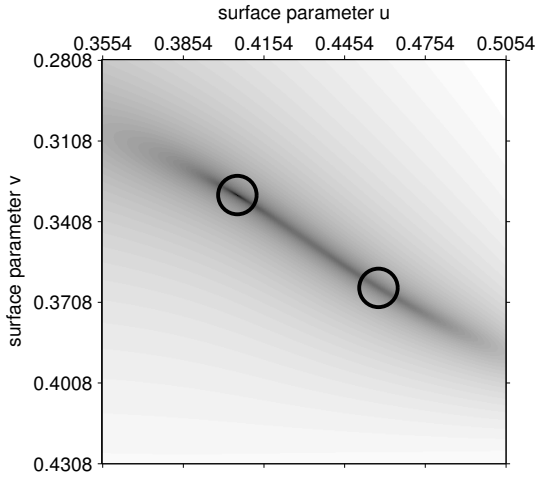
Figure 4.3.1.: Cost function of the forward projection procedure for two different camera models.

the caustic center of the camera.

The cost function $m'(u, v, \boldsymbol{P}_{cam_e})$ should now have a single minimum at parameter position $(u_e, v_e)^T$. Figure 4.3.1a indicates that this is true for a pinhole camera where the surface model was generated with a data point position distance of 42 pixels and a grid of (8×8) control points. The same parameters were used to construct a surface model for a hypercatadioptric camera. Determining the values of the cost function reveals the profile shown in Figure 4.3.1b. It exhibits a second minimum at the lower right border. There, the ray parameters get closer to the second viewing ray which also coincides with $\boldsymbol{P}_{cam_e}$, but points into the opposite direction. Although this would be a valid mathematical solution for the optimization problem (4.3.5), it is no correct result for the forward projection procedure. Therefore, the orientation of the viewing ray is also taken into account, setting the cost to a high penalty value in case the ray points into a direction that differs significantly from the initial ray at sample position (u_s, v_s). In Figure 4.3.2a, the cost function for a surface model with $d_d = 4$ and $n + 1 = 12$ of a conic camera is shown. Here, the lower right side has a unitary high value, indicating the false direction of the corresponding viewing rays. It is also revealed that the shape of the area around the minimum is different, compared to those of the pinhole and hyperbolic cost functions. The more stretched thin shape can make it more difficult to get starting points that lead



(a) $m'(u, v, \boldsymbol{P}_{cam_e})$ for the surface model of a conic camera ($d_d = 4, n + 1 = 12$). $\boldsymbol{P}_{cam_e}$ was generated from pixel position $\boldsymbol{p}_{img_e} = (256, 192)^T$.



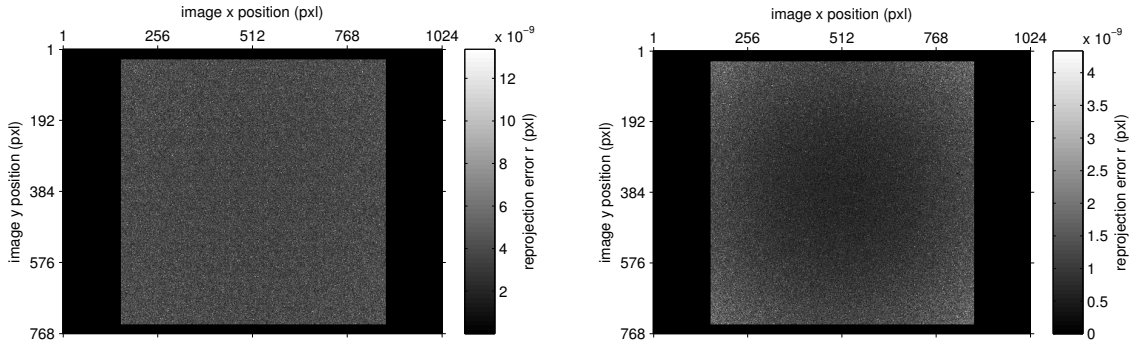
(b) $m'(u, v, \boldsymbol{P}_{cam_e})$ for the surface model of a conic camera ($d_d = 4, n + 1 = 12$). Here, $\boldsymbol{P}_{cam_e}$ was generated from pixel position $\boldsymbol{p}_{img_e} = (440, 255)^T$. This leads to a thin valley with two local minima (marked by circles). The left circle lies at the correct global minimum.

Figure 4.3.2.: Cost function of the forward projection procedure for a conic camera with two different $\boldsymbol{P}_{cam}$.

to fast convergence. This becomes especially relevant when there are multiple minima in close proximity within this valley. Figure 4.3.2b shows the cost function for a conic camera with $\boldsymbol{P}_{cam_e}$ lying on the viewing ray of pixel position $(440, 255)^T$. There are actually two local minima in close vicinity, making it difficult to find the global one. Increasing the number of sample points $(u_{si}, v_{si})^T$ by decreasing there mutual distances is a possibility to solve this problem.

To test the accuracy of the forward projection procedure, a point $\boldsymbol{P}_{cam_e}$ is generated for every pixel position in the modeled area and subsequently forward projected to the image plane. The distance between the resulting image position $\boldsymbol{p}_{img_f}$ and the original pixel position $\boldsymbol{p}_{img_e}$ is called *reprojection error* and serves as a measure for the accuracy. For the pinhole and the hypercatadioptric camera, the error is negligible, lying at $3.95 \cdot 10^{-9}$ and $1.09 \cdot 10^{-9}$ pixels on average, respectively. The corresponding profiles are shown in Figures 4.3.3a and 4.3.3b. These remaining errors depend only on the selected convergence criteria of the Nelder-Mead algorithm. Furthermore, the accuracy is mostly independent of the image location.

The situation slightly changes for the conic camera. As illustrated in Figure 4.3.2b, multiple local minima that lie close together make it difficult to identify the correct one. When keeping the amount of sample points that are used to determine good starting points for



(a) Reprojection error profile for a pinhole surface model, generated with $d_d = 42$ and $n+1 = 8$. (32×32) equally distributed sample points are used to determine the best starting point for the nonlinear optimization procedure.

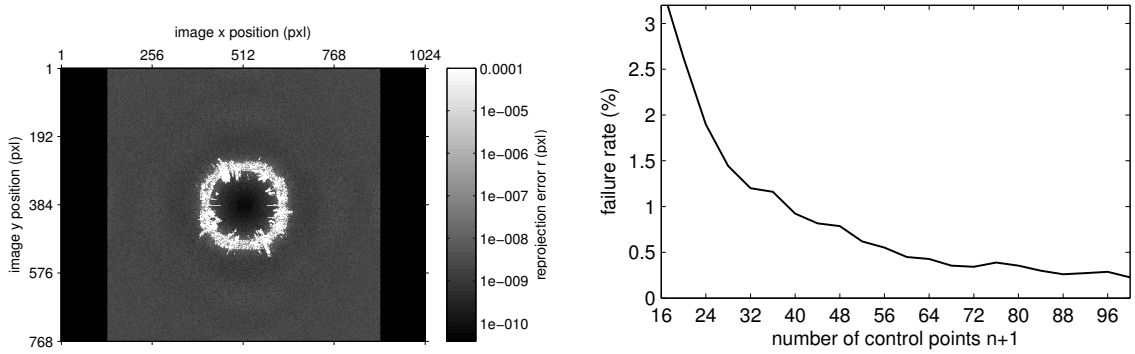
(b) Reprojection error profile for a hypercatadioptric surface model, generated with $d_d = 42$ and $n + 1 = 8$. A grid of (32×32) sample points is used for starting point determination.

Figure 4.3.3.: Profiles of the reprojection error for two different camera models.

the optimization procedure at (32×32) as before, it happens that not the desired global minimum is found. The procedure is said to have failed when the reprojection error is bigger than 1 pixel. In the current example of the conic camera ($d_d = 4$, $n + 1 = 12$), this is the case for 3.2% of all processes. The average reprojection error of all succeeded procedures is $3.22 \cdot 10^{-4}$ pixels. Figure 4.3.4a illustrates this. Positions of unsuccessful forward projections are marked in white. To decrease the failure rate, the number of spline control points of the surface model can be increased. When (40×40) points are used, the failure rate drops below 1%. Figure 4.3.4b shows a plot which illustrates the development of the failure rate depending on the number of control points.
Despite these difficulties, the proposed method is a good choice for general forward projection with the surface model. The presented cost function is locally convex and the Nelder-Mead optimization algorithm in general converges quickly and robustly. In case of convergence, the remaining residual lies within a nanopixel. The procedure is therefore more than sufficiently accurate for common measurement purposes.

4.3.1. General forward projection of uncertain points

The last section showed how general forward projection can be achieved with the spline surface model. In that case, the position of the projected point $\boldsymbol{P}_{cam}$ was perfectly known. However, it is useful for practical applications to be able to project also points that are uncertain. These occur for example when the point position was determined by intersecting two (or multiple) uncertain viewing rays. Or, as it is the case during the calibration



(a) Reprojection errors for the surface model of a conic camera ($d_d = 4$, $n + 1 = 12$, (32×32) sample points). Positions where the projection procedure failed are displayed in white.

(b) Increasing the number of control points for the surface model decreases the failure rate.

Figure 4.3.4.: Analysis of the reprojection error for the conic camera. Multiple minima can make it difficult to identify the correct solution.

procedure described in Chapter 5, when a point lies on an uncertain plane.
Due to the piecewise definition of the spline surface model it is impossible to execute an analytical uncertainty propagation procedure. The reason is that points which lie apart from the mean position might be projected to a different section of the spline surface which has different internal parameters. This means that the analytical description of the projection procedure can vary, depending on the distance of a sample point from the distribution mean. These variations cannot be described by a single mapping and prevent analytical uncertainty propagation. For this reason, the unscented transform (Section 2.2.4) will be used to determine the uncertainty of the forward projection result.
The task is to determine the pixel $\boldsymbol{p}_{img}$ and its covariance matrix $\Sigma_{p_{img}p_{img}}$ as a result of the forward projection of a point that is a normally distributed random variable in 3D space with mean value $\boldsymbol{P}_{cam}$ and covariance matrix $\Sigma_{\boldsymbol{P}_{cam}\boldsymbol{P}_{cam}}$. Following the procedure of the unscented transform, first the sigma points $\boldsymbol{P}_{cam,s}$, $s = 0..6$, are generated via

$$\begin{aligned}
\boldsymbol{P}_{cam,0} &= \boldsymbol{P}_{cam} \\
\boldsymbol{P}_{cam,i} &= \boldsymbol{P}_{cam} + \sqrt{3}\boldsymbol{s}_i, \quad i = 1..3 \\
\boldsymbol{P}_{cam,3+i} &= \boldsymbol{P}_{cam} - \sqrt{3}\boldsymbol{s}_i, \quad i = 1..3
\end{aligned}$$

where $\boldsymbol{s}_i$ is the ith column of $\sqrt{\Sigma_{P_{cam}P_{cam}}}$. The value for κ was chosen to be $d - 3 = 0$ as proposed. This leads to $d + \kappa = 3$ and the associated weights $\mathcal{W}_0 = 0$ and $\mathcal{W}_{1..6} = 1/6$. Subsequently, the sigma points are forward projected by the procedure described in the last section to achieve the transformed points $\boldsymbol{p}_{img,s}$. The corresponding covariance matrix

$\Sigma_{p_{img}p_{img}}$ is determined as shown in (2.2.33) via

$$\Sigma_{p_{img}p_{img}} = \frac{1}{6}\sum_{s=1}^{6}(\boldsymbol{p}_{img,s} - \boldsymbol{p}_{img,0})(\boldsymbol{p}_{img,s} - \boldsymbol{p}_{img,0})^T. \tag{4.3.9}$$

To test the approach, a surface model of a hyperbolic camera is used. As before, points $\boldsymbol{P}_{cam_e}$ in camera coordinates are generated by choosing positions on the viewing rays of each image pixel at a distance of 10 meters. Their uncertainty is described by a covariance matrix $\Sigma_{P_{cam}P_{cam}} = \mathrm{I}_3\sigma^2$. A Monte Carlo simulation is executed by drawing 10.000 samples from the corresponding normal distribution, forward projecting them to the image plane and determining the covariance matrix $\Sigma^{MC}_{p_{img_e}p_{img_e}}$ of the results. This is compared to the output $\Sigma^{UT}_{p_{img_e}p_{img_e}}$ of the unscented transform by determining two ratios which express the similarity of two covariance matrices. The first one uses the standard deviations σ_1 and σ_2 along the main axes of the covariance ellipses to determine the two values

$$r_1 = \frac{\min(\sigma_1^{UT}, \sigma_1^{MC})}{\max(\sigma_1^{UT}, \sigma_1^{MC})} \quad \text{and} \quad r_2 = \frac{\min(\sigma_2^{UT}, \sigma_2^{MC})}{\max(\sigma_2^{UT}, \sigma_2^{MC})}.$$

Their average builds the first ratio for uncertainty comparison

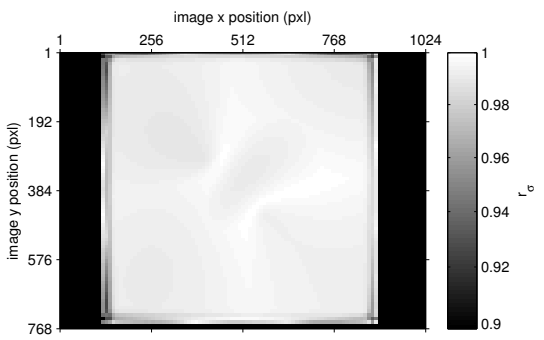
$$r_\sigma = \frac{r_1 + r_2}{2}. \tag{4.3.10}$$

The closer r_σ comes to 1, the more alike the covariance ellipses are. As also the main direction of the uncertainties is of importance, a second ratio is calculated which gives a measure for the similarity of the orientation. Given the angles α^{UT} and α^{MC} (in degrees) of the main axes of the covariance matrices, the following value is determined:

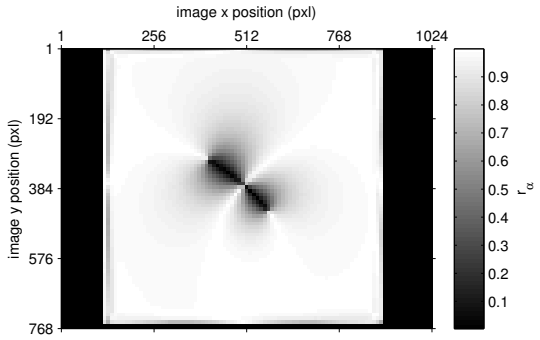
$$r_\alpha = 1 - \frac{|\alpha^{UT} - \alpha^{MC}|}{90}. \tag{4.3.11}$$

As before, similar covariance matrices lead to values close to 1 for r_α. Figures 4.3.5a and 4.3.5b show the resulting profiles for the hyperbolic camera. A comparison of selected covariance ellipses can be seen in Figure 4.3.6. In general, the results of the Monte Carlo simulation and the unscented transform are very similar. The only remarkable difference can be observed for angles at positions close to the image center. But a consideration of the corresponding ellipses reveals, that they are close to circular in this region. Therefore, small values of r_α do not imply big differences between the ellipses in this case. In fact, the uncertainty distributions are still very similar, as can be inferred from a visual comparison of the ellipses. This proves that the proposed procedure can indeed be used to achieve general forward projection of an uncertain point with the surface model.

This concludes the chapter on the continuous representation of the generic camera model. It was shown in Section 4.1 how a spline surface in 6D Plücker space, the so-called *surface*



(a) Standard deviation ratio r_σ for the forward projection procedures of an uncertain point for each pixel (see text for explanations).



(b) Angular ratio r_α for all pixels. Small values close to the center are a consequence of the circular covariance ellipses in that region (confer text and Figure 4.3.6).

Figure 4.3.5.: Forward projection of an uncertain point with a surface model of a hyperbolic camera. Compared are the covariance matrices obtained from Monte Carlo estimation and the unscented transform.

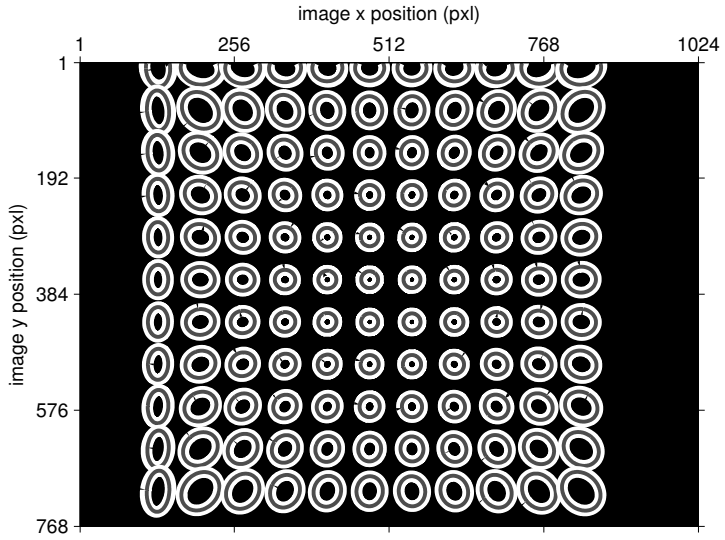


Figure 4.3.6.: The pixel distributions after forward projecting of an uncertain point with a hyperbolic camera. White ellipses mark the 2σ boundary of the results of the Monte Carlo procedure. The uncertainties from the unscented transform are visualized in gray.

model, can be used to describe arbitrary (non-central) cameras. Section 4.2 proved that a high modeling accuracy can be achieved. In Section 4.2.2 it was shown how uncertain data points can be used to create uncertain spline curves and surfaces. This lead to the uncertain surface model presented in Section 4.2.3 which provides viewing ray parameters and an associated covariance matrix for every camera pixel. Section 4.3 introduced a method for general forward projection of arbitrary points to the image plane, utilizing a non-linear optimization procedure to find the minimum of a specified cost function. Finally, Section 4.3.1 showed how the forward projection method can be augmented to handle uncertain 3D points and provide a covariance matrix together with the pixel position.

All experiments in this chapter were based on simulations. The next chapter will show how real cameras can be calibrated, i.e. how the data points necessary to build the surface model can be determined from images of calibration objects.

Chapter 5

Calibration of the surface model

In the previous chapter, the surface model was introduced as an uncertain continuous representation of the generic camera model. It was shown that a spline surface in 6D Plücker space can be used to describe the mappings from pixel position to viewing ray (back projection) as well as from 3D points to the digital image (forward projection) with high accuracy. The corresponding covariance matrices of the line parameters and the pixel position provide the user with uncertainty information.
The current chapter introduces a calibration procedure for this surface model. Some global settings of the spline surface are fixed: the knot parameterization is always uniform and the basis functions are set to be cubic (see Section 2.3 for more information on B-Splines). Others, like the data point position distance d_d and the number of control points $n + 1$ may be adjusted according to the complexity of the camera mapping function (compare Section 4.2.1). The focus of this chapter lies on the determination of the data points needed for building the spline surface. They lead to the local parameters of the surface model, namely the positions of the control points and their uncertainties. In fact, the construction of the spline surface itself is only the last step of the calibration procedure. It will be described in Section 5.3.

Determining the parameters of the generic model is much more cumbersome than for standard models, due to their sheer number. A lot more measurements are necessary to gather the desired information. Making this process as easy as possible for the user is one of the main goals when devising the calibration procedure described in this work. The most convenient method proposed so far is the one by Sturm and Ramalingam in [SR04]. They utilize images of hand-held sparse planar calibration patterns, which requires a minimum of preparation. It also spares the user from creating 3D structures with high accuracy or using complex methods such as coded light approaches as done by Grossberg and Nayar in [GN01] and Dunne et al. in [DMW10]. Generally, further equipment like digital dis-

plays or external tracking devices, as e.g. utilized by Miraldo and Araujo [MA11, MA13] to determine the plane poses, should be avoided.
For these reasons, a hand-held calibration plane is also used in this work as the only equipment for calibration. A chessboard pattern on this plane provides the visual landmarks that define fixed positions in the local coordinate system. The corresponding chessboard corners can be located in camera images with high accuracy by using dedicated methods of digital image processing. These chessboard corner positions in the camera image are the only real measurements taken for the whole calibration procedure. All other information is inferred from them, as are the corresponding uncertainties.
One of the reasons for the high complexity of generic camera calibration is that multiple measurements are needed for every single pixel. Commonly, coded light approaches are used to achieve this. When sparse calibration patterns like chessboards are utilized, only singular measurements are directly available. Therefore, an interpolation method is necessary to infer information on the local calibration plane position for pixels where actually no data has been gathered. Sturm and Ramalingam use homographic interpolation in [SR04] which locally approximates the camera mapping with a linear function. The experiment conducted in [RW12b] shows that this induces great inaccuracies for imaging systems with severe non-linear distortions. Dunne's comparison of calibration methods in [DMW07] indicates that these deviations have a negative influence on the overall performance of the procedure, as their results with active grids are much better than those achieved with homographic interpolation. In [RW12b] it is shown that the interpolation abilities of B-spline surfaces exceed those of the homographic approach. For this reason, B-spline surfaces are used in this work for interpolation purposes and are utilized to generate intermediate measurements. The exact procedure is described in Section 5.1.1. To summarize, Table 5.0.1 shows a comparison of the mentioned calibration procedures and their properties.

The calibration procedure proposed in this work comprises multiple steps. They can roughly be divided into three phases: *initial calibration*, *complete calibration* and *surface construction*:

1. In the initial calibration phase (Section 5.1), first the poses of a few selected calibration planes are determined by a linear approach. They are refined with a bundle adjustment procedure and lead to a partly calibrated camera image.

2. The next phase (Section 5.2) leads to a complete calibration of the camera. This is achieved by iteratively adding further planes to expand the calibrated area. Bundle adjustment is used for refinement.

3. In the last phase (Section 5.3), at first line parameters are determined in Plücker coordinates from the calibration planes for specified pixel positions. They are then used as data points to finally build the surface model for the camera.

Authors	Equipment	HH	NC	U
Grossberg and Nayar 3.3.2	electronic display, coded light approach, turntable	no	yes	no
Sturm and Ramalingam 3.3.3	plane with sparse pattern IF: homography	yes	(yes)*	no
Dunne et al. 3.3.4	electronic display, coded light approach	yes	no	no
Miraldo and Araujo 3.3.5	plane with sparse pattern, external position tracking IF: radial basis functions	yes	yes	no
Rosebrock and Wahl [RW12b, RW12a]	plane with sparse pattern IF: B-spline surface	yes	yes	no
this work	plane with sparse pattern IF: B-spline surface	yes	yes	yes

Table 5.0.1.: Comparison of calibration procedures for the generic camera model. IF=interpolation function, HH=hand-held (freely positioned) calibration objects, NC=works with non-central cameras, U=incorporates uncertainties. *Sturm's approach works for central and non-central cameras, but only if the type is known in advance.

During all these steps, measurement uncertainties are explicitly propagated to obtain an uncertain surface model in the end. The model allows to generate ray parameters for arbitrary pixel positions. The corresponding covariance matrices will serve as weights during the bundle adjustment procedure described in Section 5.1.6. Simulations will prove the applicability of the proposed concepts. Results for various real cameras will be shown in Chapter 6.

5.1. Initial calibration

Calibrating the generic camera model as proposed by Grossberg and Nayar in [GN01] means to determine the viewing ray parameters for every pixel. Therefore, at least two measurements, i.e. points in 3D space, have to be obtained for each of them. In this work, these points are the intersections of the viewing rays with freely placed calibration planes. The positions of these intersections in the local plane coordinate systems can be determined directly from the taken calibration images, either via a coded light approach or by using an interpolation procedure in case sparse calibration patterns are used. Therefore, the remaining task of the calibration procedure is to obtain the poses of the calibration planes in a common coordinate system. For central systems without lens distortions,

which have a linear mapping function, multiple view geometry is commonly used to gain these poses. However, for systems with distortions, possibly being omnidirectional and non-central, the corresponding laws cannot be applied. Dunne et al. [DMW10] evade this problem by restricting their calibration procedure to central cameras. The process is described in detail in Section 3.3.4. To summarize their approach: under the assumption of a central system it is possible to generate artificial perspective views from distorted images of calibration planes. This is done by using one of these planes as a virtual image plane, which also defines the camera coordinate system CCS. Subsequently, standard calibration procedures can be utilized to determine the relative poses ${}^{cam}\mathrm{T}_v$ of the remaining calibration planes v. The application of this method is theoretically justified only for central cameras. Nevertheless, experiments will show that it can be used to obtain initial values for the plane poses, even when the setup is non-central. The initial step of determining the poses from a linear equation system is followed by a non-linear *bundle adjustment* (BA) procedure which refines the results by minimizing the summed ray-point distance d_r. This distance is defined as

$$d_r = \sum_{i=1}^{N_i} \sum_{v=1}^{V} d(\boldsymbol{r}_i(x_i, y_i), \boldsymbol{P}_{cam,vi}). \tag{5.1.1}$$

Here, d is the Euclidean distance between the viewing ray $\boldsymbol{r}_i$ of pixel i and 3D point $\boldsymbol{P}_{cam,vi}$, which is the local position $\boldsymbol{p}_{vi}$ on plane v seen by pixel i and transferred to the camera coordinate system. An illustration can be found in Figure 3.3.5. The bundle adjustment also compensates for the errors that occur in the initial plane pose estimates when the camera is non-central.
The following sections will first cover the theoretical basics needed to execute the initial calibration procedure, namely the spline surface interpolation in Section 5.1.1 and the determination of homographies with uncertain points in Section 5.1.2. The execution of the linear calibration procedure, also with uncertain inputs, will be described in Section 5.1.4. Subsequently, in Section 5.1.6, the bundle adjustment procedure is introduced and applied to get the final results of the initial calibration process.

5.1.1. Interpolation on sparse calibration patterns

The ultimate goal of this chapter is to determine viewing rays for arbitrary image pixel positions which will then be used as data points to build the surface model. As mentioned in the introduction, the only measurements taken are the pixel positions of the corners of a sparse chessboard calibration pattern. The corresponding metric locations in the local plane coordinate system are known, defining a relation between sparse image coordinates and points in the real world. To get measurements for arbitrary pixels, i.e. to determine the local position where a specified viewing ray $\boldsymbol{r}_i$ intersects the calibration plane, an interpolation method is needed. This basic function will be needed several

times along the calibration process and will therefore be described in detail in this section.

The chessboard corner positions in the image serve as data points $\boldsymbol{p}_{img_c}$ used to build a surface $\boldsymbol{S}_c(u,v)$. In this case, the parameters u and v are linearly related to the 2D calibration plane positions. The procedure introduced in Section 4.2.2 allows to integrate the measurement uncertainties $\Sigma_{p_c p_c}$ of the chessboard corner pixel positions. Considering the spline surface, the pixel position is the data domain and the metric calibration plane position is represented by the spline parameters u and v. The task is now to determine the parameters $(u_p, v_p)^T$ and the corresponding covariance matrix $\Sigma_{uu,vv}$ for an arbitrary image position. This procedure is called uncertain spline inversion and it will be described in the next subsection how it can be executed for a general uncertain spline surface.

5.1.1.1. Inversion of uncertain spline surfaces

The procedure of general spline inversion was introduced in Section 2.3.1. It determines the parameter value u_p (or the values $(u_p, v_p)^T$ for a surface) of that point of the spline which is closest to a given point $\boldsymbol{P}$. $\boldsymbol{P}$ itself does not necessarily lie on the spline. An iterative procedure was proposed, which is applicable to spline curves and surfaces with arbitrary degree p. When the spline surface is created with uncertain data points, the control points are also uncertain, as is the result of the inversion process. This section will show how the corresponding covariance matrix can be determined for a general uncertain spline surface. The inversion point $\boldsymbol{P}$ is considered to be perfectly known, i.e. it is free of any uncertainties.
It is assumed that the correct surface parameters $(u_p, v_p)^T$ are already determined by the iterative procedure from Section 2.3.1. From (2.3.11) and (2.3.12) it follows that to following condition is fulfilled:

$$\begin{bmatrix} f_s(u_p, v_p, \boldsymbol{P}, \boldsymbol{P}_{00}, ..., \boldsymbol{P}_{nm}) \\ g_s(u_p, v_p, \boldsymbol{P}, \boldsymbol{P}_{00}, ..., \boldsymbol{P}_{nm}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.1.2}$$

$$\Leftrightarrow \quad \begin{bmatrix} (\boldsymbol{S}(u,v) - \boldsymbol{P})^T \\ (\boldsymbol{S}(u,v) - \boldsymbol{P})^T \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_u(u,v) \\ \boldsymbol{S}_v(u,v) \end{bmatrix} \Bigg|_{\substack{u=u_p \\ v=v_p}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.1.3}$$

$$\Leftrightarrow \quad \begin{bmatrix} (\boldsymbol{S}(u_p,v_p) - \boldsymbol{P})^T \\ (\boldsymbol{S}(u_p,v_p) - \boldsymbol{P})^T \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_u(u_p,v_p) \\ \boldsymbol{S}_v(u_p,v_p) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{5.1.4}$$

This implicit function contains the surface parameters u_p and v_p as well as the control points $\boldsymbol{P}_{ij}$ and the first partial derivatives of the spline surface in u and v direction, $\boldsymbol{S}_u$ and $\boldsymbol{S}_v$. By using the covariance matrices of the control points, the (2×2) covariance matrix of the corresponding parameters are to be calculated. The law of implicit uncertainty propagation (see section 2.2.3) allows to do so, without having to determine explicit

formulations for u_p and v_p. It states that

$$\Sigma_{u,v} = \mathrm{B}^{-1}\mathrm{A}\Sigma_{\boldsymbol{P}_p\boldsymbol{P}_p}\mathrm{A}^T\mathrm{B}^{-T}. \tag{5.1.5}$$

Here, $\Sigma_{\boldsymbol{P}_p\boldsymbol{P}_p}$ is the covariance matrix of the involved control points $\{\boldsymbol{P}_{s_u-p,s_v-p}, ..., \boldsymbol{P}_{s_u,s_v}\}$ with s_u and s_v being the first relevant indices for the current parameter position $(u_p, v_p)^T$ and p being the degree of the basis functions. Matrix A contains the partial derivatives for all elements of these d-dimensional control points:

$$\mathrm{A} = \begin{bmatrix} \frac{\partial f_s}{\partial P_{s_u-p,s_v-p,0}} & \frac{\partial f_s}{\partial P_{s_u-p+1,s_v-p,0}} & \cdots & \frac{\partial f_s}{\partial P_{s_u,s_v,0}} & \frac{\partial f_s}{\partial P_{s_u-p,s_v-p,1}} & \cdots & \frac{\partial f_s}{\partial P_{s_u-p,s_v-p,d}} \\ \frac{\partial g_s}{\partial P_{s_u-p,s_v-p,0}} & \cdots & \cdots & \cdots & \cdots & \cdots & \frac{\partial g_s}{\partial P_{s_u-p,s_v-p,d}} \end{bmatrix}. \tag{5.1.6}$$

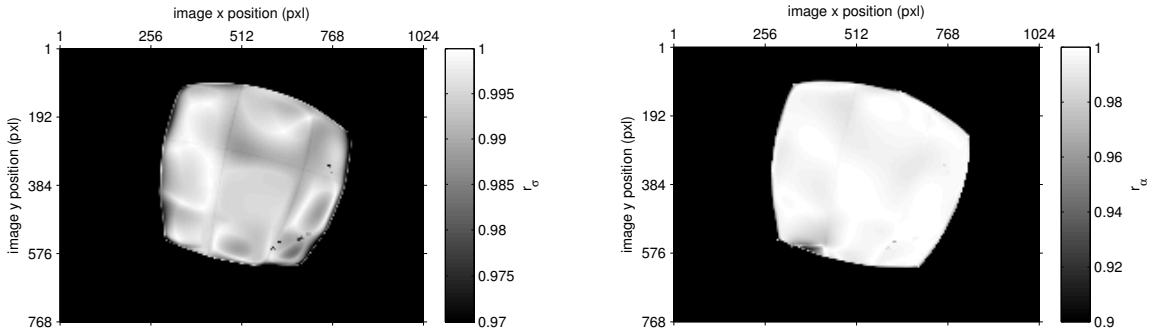
Matrix B comprises of the partial derivatives of the parameters:

$$\mathrm{B} = \begin{bmatrix} \frac{\partial f_s}{\partial u_p} & \frac{\partial f_s}{\partial v_p} \\ \frac{\partial g_s}{\partial u_p} & \frac{\partial g_s}{\partial v_p} \end{bmatrix}. \tag{5.1.7}$$

A detailed derivation of A and B can be found in A.1.
For the current application, $\boldsymbol{S}(u, v)$ is a spline surface in the 2D camera image. It is created by using the uncertain chessboard corner positions $\boldsymbol{P}_{ij} = \boldsymbol{p}_{img_{c,ij}}$ and their covariance matrices $\Sigma_{\boldsymbol{p}_{c,ij}\boldsymbol{p}_{c,ij}}$. The inversion point is an arbitrary pixel position in the camera image $\boldsymbol{P} = \boldsymbol{p}_{img_a}$ which is free of any uncertainties. A short experiment will prove the applicability of the proposed procedure: an uncertain spline surface like the one depicted in Figure 4.2.25 is generated with the standard deviation of the data point noise varying around 5 pixels in x and y direction. The results of the uncertainty propagation $\Sigma_{u,v}^{UP}$ calculated by (5.1.5) are then compared to those obtained via Monte Carlo simulation $\Sigma_{u,v}^{MC}$. Figure 5.1.1 shows the profiles of the standard deviation ratio r_σ and the angular ratio r_α (which are defined equivalently to (4.3.10) and (4.3.11)). It can be seen that there is a high compliance. This justifies the use of the described procedure of uncertain spline surface inversion for determining the covariance matrices of the parameter positions.

5.1.2. Homography from uncertain points

The linear calibration procedure that will be used to get initial values for the plane poses is described in detail in Section 3.1.1. In a first step, the homographies between the virtual camera image and the other calibration views need to be determined. Generally, this can be done by using the *direct linear transform* (DLT) algorithm described by Hartley and Zisserman in [HZ03]. It utilizes data normalization (commonly called *conditioning* in linear algebra) and singular value decomposition to obtain a stable solution. However, that method does not take the uncertainties of the used points into account. In this work, all points are uncertain. Therefore, the procedure proposed by Förstner in [FW14], will



(a) Standard deviation ratio r_σ for the 2D spline surface inversion procedure.

(b) Angular ratio r_α for the spline inversion procedure.

Figure 5.1.1.: Inversion of an uncertain spline surface. Comparison of the covariance matrices $\Sigma_{u,v}^{MC}$ from a Monte Carlo estimation with those from a first order uncertainty propagation $\Sigma_{u,v}^{UP}$ for each pixel.

be applied to determine the homographies and the covariance matrices of their elements. Here, the homographies are determined for a virtual camera that has one of the calibration planes as its image plane, which is denoted by the view index $v = 1$. Therefore, each homography H_v is a mapping from local point position $\mathbf{p}_v = (x_v, y_v, 1)^T$ on a second calibration plane v to point positions $\mathbf{p}_1 = (x_1, y_1, 1)^T$ on the virtual image plane:

$$\mathbf{p}_1 = \mathrm{H}_v \mathbf{p}_v. \tag{5.1.8}$$

The data points that are needed to calculate H_v are determined in the following way:

1. Select pixel positions $\boldsymbol{p}_{img,i} = (x_{img,i}, y_{img,i})^T$, $i = \{1...N_i\}$, which lie inside the area covered by the calibration pattern of the virtual image plane and the second calibration plane v.

2. Determine the local positions $\boldsymbol{p}_{1,i}$ and $\boldsymbol{p}_{v,i}$ where ray i intersects the two calibration planes. Here, the patterns on the planes are sparse chessboards, which do not deliver distinct visual cues for every single pixel. Therefore, the local positions for the selected pixels $\boldsymbol{p}_{img,i}$ are determined by fitting a B-spline surface through the uncertain chessboard corners and executing the spline inversion procedure described in Section 2.3.1. This delivers the spline parameters $(u_{1,i}, v_{1,i})^T$ and $(u_{v,i}, v_{v,i})^T$ which are linearly connected to the local plane positions. Multiplying them with the widths and heights of the calibration patterns then leads to the desired metric plane coordinates $\boldsymbol{p}_{1,i}$ and $\boldsymbol{p}_{v,i}$.

3. To determine the corresponding covariance matrices $\Sigma_{\boldsymbol{p}_{img,i}\boldsymbol{p}_{img,i}}$ for each point $\boldsymbol{p}_{img,i}$ separately, the procedure described in Section 5.1.1.1 can be used. It is applicable

because the pixel positions are selected directly and therefore free of any uncertainties. The obtained parameter covariances Σ_{uu,vv_i} subsequently also need to be scaled by the width and height of the current calibration pattern.
In the current application, the inversion procedure is executed for multiple points on the same spline surface v. This allows to take the covariances between all involved spline control points into account, i.e. to use the full covariance matrix $\Sigma_{\boldsymbol{PP}_v}$ for all elements of every control point. The matrices A_v and B_v needed for uncertainy propagation can be adjusted accordingly. Then, the modified version of (5.1.5) can be used to obtain the full covariance matrices Σ_{uu,vv_f} of all parameter positions. Scaling with width and height of the calibration pattern delivers the final full matrix $\Sigma_{\boldsymbol{pp}_v}$ for every element of all plane points.

The generated point correspondences $\boldsymbol{p}_{1,i}$ and $\boldsymbol{p}_{v,i}$ ($i = \{1...N_i\}$) and the according full covariance matrices $\Sigma_{\boldsymbol{pp}_1}$ and $\Sigma_{\boldsymbol{pp}_v}$ need to be transferred to their homogeneous equivalents $\mathbf{p}_{1,i}$, $\mathbf{p}_{v,i}$, $\Sigma_{\mathbf{pp}_1}$ and $\Sigma_{\mathbf{pp}_v}$. Then, the homography H_v and the covariance matrix $\Sigma_{\boldsymbol{hh}_v}$ of its elements can be determined in the following way. The model function (5.1.8) has to be fulfilled for every point pair i:

$$\mathbf{p}_{1,i} = \mathrm{H}_v \mathbf{p}_{v,i}. \tag{5.1.9}$$

This can be converted to different implicit formulations:

$$\begin{aligned} \boldsymbol{f}_i(\mathbf{p}_{1,i}, \mathbf{p}_{v,i}, \boldsymbol{h}_v) &= \mathrm{S}'(\mathbf{p}_{1,i})\mathbf{p}_{1,i} = \mathbf{0} \\ &\Leftrightarrow \quad \mathrm{S}'(\mathbf{p}_{1,i})\mathrm{H}_v\mathbf{p}_{v,i} = \mathbf{0} && (5.1.10) \\ &\Leftrightarrow \quad \mathrm{S}'(\mathbf{p}_{1,i})(\mathbf{p}_{v,i}^T \otimes \mathrm{I}_3)\boldsymbol{h}_v = \mathbf{0} && (5.1.11) \\ &\Leftrightarrow \quad -\mathrm{S}'(\mathrm{H}_v\mathbf{p}_{v,i})\mathbf{p}_{1,i} = \mathbf{0}. && (5.1.12) \end{aligned}$$

$\mathrm{S}(\mathbf{p})$ is the skew symmetric matrix of a vector $\mathbf{p}$ which replaces the cross product. The primed notation $\mathrm{S}'(\mathbf{p})$ indicates that it is restricted to its first two rows to guarantee linear independence of the equations (which are used later to build an equation system). $\boldsymbol{h}_v$ is a vector containing the stacked columns of H_v and $\otimes$ symbolizes the Kronecker product of two matrices.
To determine H_v, first a matrix A is created by stacking (5.1.11) from $N_i \geq 4$ point pairs:

$$\underset{(2N_i \times 9)}{\mathrm{A}} = \begin{bmatrix} \mathrm{S}'(\mathbf{p}_{1,1})(\mathbf{p}_{v,1}^T \otimes \mathrm{I}_3) \\ \vdots \\ \mathrm{S}'(\mathbf{p}_{1,N_i})(\mathbf{p}_{v,N_i}^T \otimes \mathrm{I}_3) \end{bmatrix} = \begin{bmatrix} \mathrm{A}_1 \\ \vdots \\ \mathrm{A}_{N_i} \end{bmatrix}. \tag{5.1.13}$$

Subsequently, the resulting equation system

$$\mathrm{A}\boldsymbol{h}_v = \mathbf{0} \tag{5.1.14}$$

is solved to determine the elements of H_v. This can be done by using the singular value decomposition and taking the singular vector belonging to the smallest singular value as

a solution for $\boldsymbol{h}_v$. The corresponding covariance matrix $\Sigma_{\boldsymbol{h}_v\boldsymbol{h}_v}$ is determined via the law of implicit uncertainty propagation (compare (2.2.15)). Here, the covariances of the data points are known, which gives the combined covariance matrix

$$\underset{(6N_i\times 6N_i)}{\Sigma_{\mathbf{pp}}} = \begin{bmatrix} \Sigma_{\mathbf{pp}_1} & 0 \\ 0 & \Sigma_{\mathbf{pp}_v} \end{bmatrix}. \tag{5.1.15}$$

Furthermore, the matrices

$$\underset{(2\times 9)}{\mathrm{A}_i} = \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{h}_v} \tag{5.1.16}$$

$$\underset{(2\times 3)}{\mathrm{B}_i} = \frac{\partial \boldsymbol{f}_i}{\partial \mathbf{p}_{1,i}} \overset{(5.1.12)}{=} -\mathrm{S}'(\mathrm{H}_v\mathbf{p}_{v,i}) \tag{5.1.17}$$

$$\underset{(2\times 3)}{\mathrm{C}_i} = \frac{\partial \boldsymbol{f}_i}{\partial \mathbf{p}_{v,i}} \overset{(5.1.10)}{=} \mathrm{S}'(\mathbf{p}_{1,i})\mathrm{H}_v \tag{5.1.18}$$

can be combined to form

$$\underset{(2N_i\times 9)}{\mathrm{A}} = \begin{bmatrix} \mathrm{A}_1 \\ \vdots \\ \mathrm{A}_{N_i} \end{bmatrix} \quad \text{and} \quad \underset{(2N_i\times 6N_i)}{\mathrm{B}} = \begin{bmatrix} \mathrm{B}_1 & & 0 & \mathrm{C}_1 & & 0 \\ & \ddots & & & \ddots & \\ 0 & & \mathrm{B}_{N_i} & 0 & & \mathrm{C}_{N_i} \end{bmatrix}. \tag{5.1.19}$$

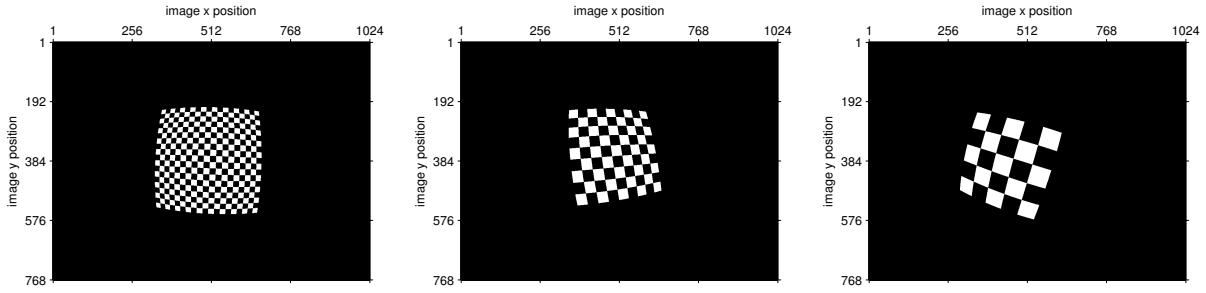
This allows to calculate the desired

$$\Sigma_{\boldsymbol{h}_v\boldsymbol{h}_v} = \mathrm{A}^{-1}\mathrm{B}\,\Sigma_{\mathbf{pp}}\,\mathrm{B}^T\mathrm{A}^{-T} \tag{5.1.20}$$

which concludes the uncertain homography determination.
Remark: for numerical reasons it is recommended to condition the plane points. Details on this step can be found in A.2.

5.1.3. Linear calibration

This section will show how the linear calibration procedure that was introduced by Dunne et al. [DMW10], see Section 3.3.4.1, can be applied to general imaging systems. Dunne et al. show that the proposed method of linearization by defining one calibration plane as a virtual image plane is theoretically justified for central cameras. The virtual images are then generated by the laws of perspective projection and the pinhole calibration method from Section 3.1.1 can be executed to determine the relative poses ${}^1\mathrm{T}_v$ of the involved calibration planes. The main difference between Dunne et al.'s procedure and the one used in this work is the way the necessary point correspondences for the homography estimation are gained. Instead of using a coded light approach (or active grids, as Dunne



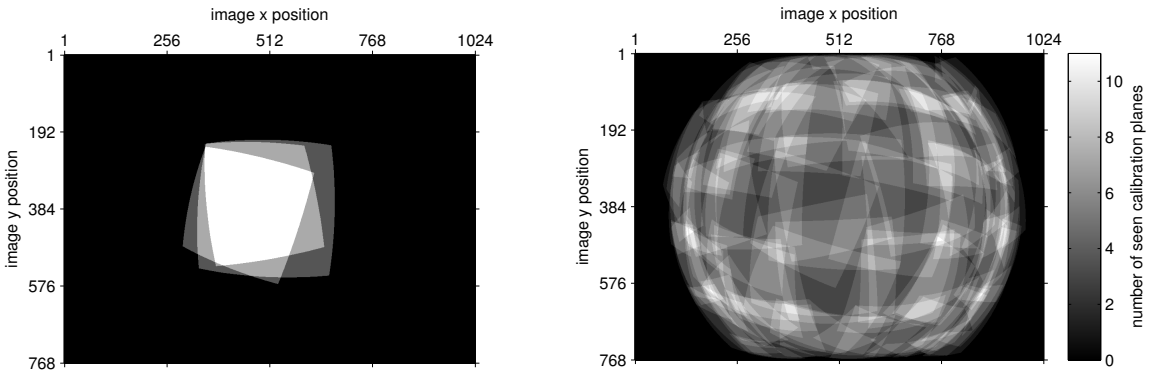
(a) (b) (c)

Figure 5.1.2.: Three exemplary images from the simulated fisheye camera. The overall size of the calibration board changes with its distance from the camera, the chessboard patches always have a size of 40x40mm.

calls them), the spline surface interpolation procedure described in Section 5.1.1 is utilized. The resulting plane points and their covariance matrices are then used to determine the homographies as shown in the previous section. The interpolation errors lead to inaccurate results for the plane poses ${}^1\mathrm{T}_v$. Two simulations with realistic parameters will therefore be executed to assess the accuracy that can be achieved, first for a central fisheye camera and subsequently for a non-central catadioptric camera with a conic mirror.

5.1.3.1. Linear calibration of a simulated fisheye camera

The simulated fisheye camera is central, has a field of view that is 190° wide and a resolution of 1024x768 pixels. Calibration plane poses are generated automatically such as three hemispheres with different radii lying in front of the camera are covered. Parallel planes lead to singular equation systems and are therefore avoided. The average distance of all planes from the optical center is 713mm and the patches of the chessboard pattern have a size of 40×40mm. Figure 5.1.2 shows some exemplary simulated camera images. The first one of them is defined as the virtual image plane and gets the number $v = 1$. As the areas in the image of the other two planes overlap with it (compare Figure 5.1.3a), the linear calibration procedure described before can be executed to get the relative plane poses ${}^1\mathrm{T}_2$ and ${}^1\mathrm{T}_3$.
Altogether, 115 calibration planes with different poses are simulated. Their overall image coverage can be seen in Figure 5.1.3b. The accuracy of the procedure is evaluated by choosing each of them separately as a virtual image plane and determining those two planes which have the biggest overlap. Then, the relative plane poses are calculated and compared to the ground truth by determining the distance $d_t = |\boldsymbol{t}_1 - \boldsymbol{t}_v|$ of the translational parts and the differences of the roll, pitch and yaw angles. Finally, the averages $\bar{d}_t$ and $\bar{d}_a$ of all position and angle errors are determined. This evaluation method is executed three



(a) Image coverage of the exemplary calibration planes from Figure 5.1.2. The overlap facilitates the use of plane 1 as virtual image plane.

(b) Image coverage of all the calibration planes generated for testing purposes.

Figure 5.1.3.: Images showing the coverage of the calibration planes for the simulated fisheye camera. Bright colors signify that more planes are seen in that area.

noise level of the $\boldsymbol{p}_{img,i}$	mean position error $\bar{d}_t$	mean angle error $\bar{d}_a$
$\sigma_p = 0.0$pxl	3.9346mm	0.0615°
$\sigma_p = 0.1$pxl	19.9083mm	0.2907°
$\sigma_p = 0.5$pxl	114.3010mm	1.7097°

Table 5.1.1.: Errors of the plane poses determined with the initial calibration procedure for a simulated fisheye camera.

times. First, the chessboard corner positions $\boldsymbol{p}_{img,i}$ in the images are perfectly known. Subsequently, they are perturbed by normal noise with standard deviations of 0.1 and 0.5 pixels. The results can be found in Table 5.1.1.
The errors of the procedure with noise-free chessboard corners are very low, lying at $\bar{d}_t = 3.9346$mm and $\bar{d}_a = 0.0615°$. For a realistic noise level of 0.1 pixels, the errors are still acceptable with $\bar{d}_t = 19.9083$mm and $\bar{d}_a = 0.2907°$. With a noise level of $\sigma_p = 0.5$pxl they already lie at $\bar{d}_t = 114.3010$mm and $\bar{d}_a = 1.7097°$. However, it has to be kept in mind that only the minimal number of 2 additional calibration planes is used here. Using more planes increases the accuracy. Furthermore, the results will subsequently be refined by a bundle adjustment procedure, as described in Section 5.1.6.

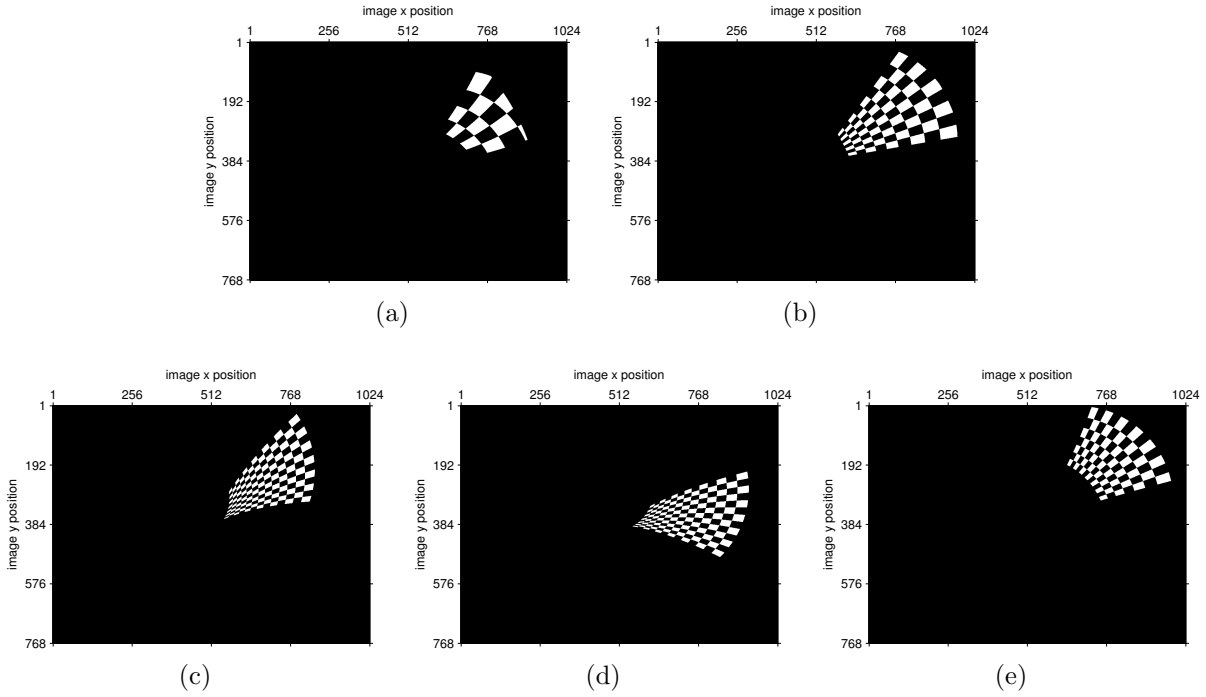
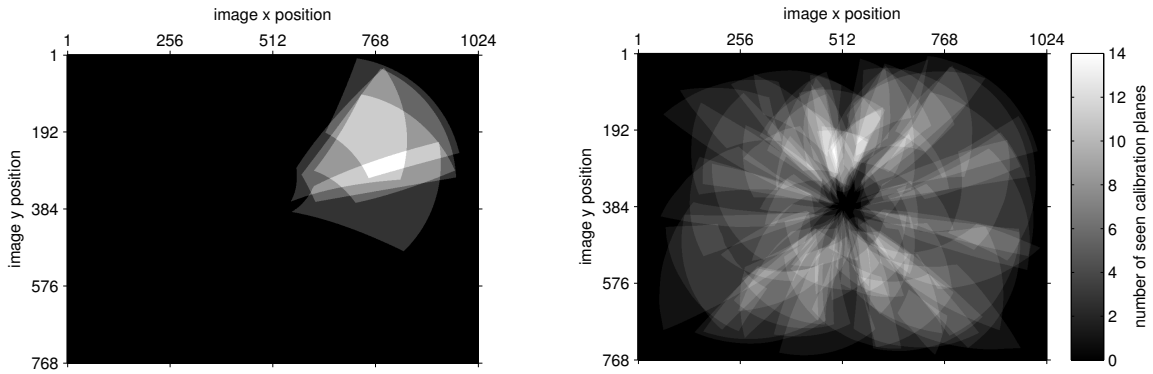


Figure 5.1.4.: Five exemplary images from the simulated catadioptric camera with a conic mirror. The size of the chessboard patches is 40x40mm.

5.1.3.2. Linear calibration of a simulated conic camera

The experiments described in the last section are repeated here for a non-central catadioptric camera with a conic mirror. The involved pinhole camera has a resolution of 1024x768 pixels and a focal length of 2mm. Its principal axis is aligned with the rotational axis of the mirror and the optical center lies 200mm away from the mirror tip. The opening angle α of the mirror is 120°. As the initial calibration procedure is based on the assumption of a central system, it becomes less robust if this condition is not met. This requires to use more than the minimum of 2 additional images. Here, each time 4 images are utilized together with the virtual image plane. Examples can be found in Figure 5.1.4. Their image coverage is shown in Figure 5.1.5a. Figure 5.1.5b illustrates the coverage of all 87 generated calibration planes. Their average distance from the optical center of the involved pinhole camera is 970mm, the patches of the chessboard pattern again have a size of 40×40mm. In spite of more images being used for the initial procedure, it sometimes still fails to get a mathematical solution (i.e. the radicand of a square root becomes negative). This happened for 3 of the 87 procedures with $\sigma_p = 0$pxl and $\sigma_p = 0.1$pxl. For $\sigma_p = 0.5$pxl, 6 of the procedures failed. Furthermore, in some cases the method does deliver a result, but the determined plane poses are comparably far from the ground truth.



(a) Image coverage of the exemplary calibration planes from Figure 5.1.4.

(b) Image coverage of all the calibration planes generated for testing purposes.

Figure 5.1.5.: Image coverages of the calibration planes for the simulated catadioptric camera with a conic mirror. Bright colors signify that more planes are seen in that area.

noise level of the $\boldsymbol{p}_{img,i}$	mean position error $\bar{d}_t$	mean angle error $\bar{d}_a$
$\sigma_p = 0.0$pxl	410.447mm	12.4711°
$\sigma_p = 0.1$pxl	419.643mm	12.8978°
$\sigma_p = 0.5$pxl	398.185mm	11.6916°

Table 5.1.2.: Errors of the plane poses determined from the initial calibration procedure for a simulated conic camera.

This leads to the high average errors shown in Table 5.1.2. Remarkably, increased noise only has a minor influence on the accuracy of these initial estimates.
These values indicate that not every plane serves equally well as a virtual image plane and not all additional planes lead to good initial estimates of the plane poses. But when the main requirements are met, i.e. the additional planes are not close to parallel and they have a large overlap with the virtual image plane, very accurate results can be achieved. These serve as good initial values for the subsequent bundle adjustment procedure described in Section 5.1.6.

5.1.4. Uncertain linear calibration

The input values of the linear method used for initial calibration are the homographies between the virtual image plane and the additional calibration planes. Section 5.1.2 shows how their covariance matrices $\Sigma_{\boldsymbol{h}_v \boldsymbol{h}_v}$ can be determined in case an estimate for the un-

certainties of the used plane points is given. These $\Sigma_{\boldsymbol{h}_v\boldsymbol{h}_v}$ can now be utilized to estimate the uncertainties of the plane poses. The initial calibration procedure incorporates several non-linear steps, including taking the square root and a matrix inversion (see (3.1.29) and (3.1.30)). This makes linearized uncertainty propagation cumbersome and inaccurate, which is why the unscented transform (Section 2.2.4) will be used to determine the plane pose uncertainties.
All elements of the homographies H_v, $v = \{2, ..., V\}$, are the input values of the procedure. This gives a total number of $d = 9(V-1)$ parameters. They are stacked in a single vector $\boldsymbol{h}_f$. The corresponding full covariance matrix $\Sigma_{\boldsymbol{h}_f\boldsymbol{h}_f}$ can be constructed from the $\Sigma_{\boldsymbol{h}_v\boldsymbol{h}_v}$ via

$$\Sigma_{\boldsymbol{h}_f\boldsymbol{h}_f} = \begin{bmatrix} \Sigma_{\boldsymbol{h}_2\boldsymbol{h}_2} & & 0 \\ & \ddots & \\ 0 & & \Sigma_{\boldsymbol{h}_V\boldsymbol{h}_V} \end{bmatrix}. \tag{5.1.21}$$

The sigma points needed for the unscented transform can be calculated as

$$\boldsymbol{h}_{f,0} = \boldsymbol{h}_f \tag{5.1.22}$$

$$\boldsymbol{h}_{f,i} = \boldsymbol{h}_f + \sqrt{d+\kappa}\boldsymbol{s}_i, \quad i = 1..d \tag{5.1.23}$$

$$\boldsymbol{h}_{f,d+i} = \boldsymbol{h}_f - \sqrt{d+\kappa}\boldsymbol{s}_i, \quad i = 1..d. \tag{5.1.24}$$

Here, $\boldsymbol{s}_i$ stands for the ith column of $\sqrt{\Sigma_{\boldsymbol{h}_f\boldsymbol{h}_f}}$, κ is set to $d-3$. These sigma points are fed to the calibration procedure, leading to the corresponding rotation vectors $\boldsymbol{r}_{1_v,i}$, $\boldsymbol{r}_{2_v,i}$, $\boldsymbol{r}_{3_v,i}$ and the plane position $\boldsymbol{t}_{v,i}$. Together with the weights $\mathcal{W}_0$ and $\mathcal{W}_i$, the covariance matrices $\Sigma_{\boldsymbol{r}_{1_v}\boldsymbol{r}_{1_v}}$, $\Sigma_{\boldsymbol{r}_{2_v}\boldsymbol{r}_{2_v}}$, $\Sigma_{\boldsymbol{r}_{3_v}\boldsymbol{r}_{3_v}}$ and $\Sigma_{\boldsymbol{t}_v\boldsymbol{t}_v}$ are determined as proposed in (2.2.33).
For subsequent optimization procedures it is convenient to have a minimal description of the plane rotations. Combining the covariance matrices of the rotational vectors to form

$$\Sigma_{\mathrm{R}_v\mathrm{R}_v} = \begin{bmatrix} \Sigma_{\boldsymbol{r}_{1_v}\boldsymbol{r}_{1_v}} & 0 & 0 \\ 0 & \Sigma_{\boldsymbol{r}_{2_v}\boldsymbol{r}_{2_v}} & 0 \\ 0 & 0 & \Sigma_{\boldsymbol{r}_{3_v}\boldsymbol{r}_{3_v}} \end{bmatrix} \tag{5.1.25}$$

allows to calculate the Rodrigues vector $\boldsymbol{R}_v$ and its covariance matrix $\Sigma_{\boldsymbol{R}_v\boldsymbol{R}_v}$ as described in Section 2.2.3.2.

The applicability of the uncertainty propagation approach will now be verified via Monte Carlo simulation. This is done by executing the initial calibration procedure with a selected set of calibration planes, first for the fisheye camera and then for the conic catadioptric camera. For both cameras, a Monte Carlo simulation with 10.000 iterations is performed. The necessary input points are generated with the help of the data vector $\boldsymbol{h}_f$ and the corresponding covariance matrix $\Sigma_{\boldsymbol{h}_f\boldsymbol{h}_f}$. The resulting covariance matrices $\Sigma^{MC}_{\boldsymbol{r}_{1_v}\boldsymbol{r}_{1_v}}$, $\Sigma^{MC}_{\boldsymbol{r}_{2_v}\boldsymbol{r}_{2_v}}$, $\Sigma^{MC}_{\boldsymbol{r}_{3_v}\boldsymbol{r}_{3_v}}$ and $\Sigma^{MC}_{\boldsymbol{t}_v\boldsymbol{t}_v}$ are compared to the outputs of the unscented transform by determining the average ratios of all their elements.

For the fisheye camera, the three calibration planes shown in Figure 5.1.2 are used, plane positions $\boldsymbol{p}_{v,i}$ are perturbed by Gaussian noise with a standard deviation of $\sigma_p = 0.1$pxl. The resulting elements of the covariance matrices of the rotational vectors agree by 81.12%. For the translational vector, a compliance of 98.24% is achieved.
Figure 5.1.4 shows the images used for calibrating the conic camera. Here, the elements of the covariance matrices agree by 34.97% and 42.55% for the rotational and the translational parts, respectively.
This shows that the results of the uncertainty propagation approach for the initial calibration do not always exactly match the Monte Carlo results. Nevertheless, the corresponding covariance matrices convey valuable information about the uncertainty of the initial plane poses and will therefore be used to form weight matrices in subsequent steps of the calibration procedure.

5.1.5. Uncertain rays from calibration planes

Each viewing ray is defined by the local plane positions $\boldsymbol{p}_{v,i}$ that are seen by the corresponding image pixel i. They define the local homogeneous points $\mathbf{P}_{v,i} = [\boldsymbol{p}_{v,i}^T \; 0 \; 1]^T$. Transferring them to the camera coordinate system by using the uncertain transformations ${}^1\mathrm{T}_v$ gives points in a common coordinate system:

$$\mathbf{C}_{v,i} = {}^1\mathrm{T}_v \mathbf{P}_{v,i}. \tag{5.1.26}$$

The covariance matrix of the inhomogeneous plane positions is defined as

$$\Sigma_{\boldsymbol{P}_{v,i}\boldsymbol{P}_{v,i}} = \begin{bmatrix} \Sigma_{\boldsymbol{p}_{v,i}\boldsymbol{p}_{v,i}} & & 0 \\ & & 0 \\ 0 \quad & 0 \quad & 0 \end{bmatrix}. \tag{5.1.27}$$

If the covariance matrices $\Sigma_{\boldsymbol{R}_v\boldsymbol{R}_v}$ and $\Sigma_{\boldsymbol{t}_v\boldsymbol{t}_v}$ of the Rodrigues vector $\boldsymbol{R}_v$, which represents the plane rotation, and of the translation vector $\boldsymbol{t}_v$ are also given, the covariance matrix $\Sigma_{\boldsymbol{C}_{v,i}\boldsymbol{C}_{v,i}}$ of the transformed point can be determined via linear uncertainty propagation (see Section 2.2.3.1 for details):

$$\Sigma_{\boldsymbol{C}_{v,i}\boldsymbol{C}_{v,i}} = \mathrm{R}_v \Sigma_{\boldsymbol{P}_{v,i}\boldsymbol{P}_{v,i}} \mathrm{R}_v^T + \Sigma_{\boldsymbol{t}_v\boldsymbol{t}_v} + \mathrm{R}_v \mathsf{S}(\boldsymbol{p}_{vi}^v) \Sigma_{\boldsymbol{R}_v\boldsymbol{R}_v} \mathsf{S}^T(\boldsymbol{p}_{vi}^v) \mathrm{R}_v^T. \tag{5.1.28}$$

The $\mathbf{C}_{v,i}$ and their $\Sigma_{\boldsymbol{C}_{v,i}\boldsymbol{C}_{v,i}}$ can then be used to execute a line fitting procedure which delivers the line parameters $\mathbf{L}_i$ and the corresponding covariance matrix $\Sigma_{\mathbf{L}_i\mathbf{L}_i}$ (see Section 2.4.4.2).
Remark: fitting a line through the points requires non-singular covariance matrices. As the virtual image plane is that calibration plane which defines the camera coordinate system, it is free of any uncertainties (i.e. $\Sigma_{\boldsymbol{R}_0\boldsymbol{R}_0} = \Sigma_{\boldsymbol{t}_0\boldsymbol{t}_0} = 0_{3\times3}$). To allow points lying on this plane to contribute to the fitting procedure, artificial non-singular covariance matrices have to be defined, e.g. by setting the diagonal elements to small values.

5.1.6. Bundle Adjustment

Section 5.1.3 showed how initial values can be obtained for the calibration plane poses by using the concept of a virtual perspective camera. Due to measurement noise on the one hand and to imaging systems that violate the assumption of centrality on the other, the results need to be refined.
In a perfectly known configuration, the viewing ray of a pixel i passes exactly through all corresponding local plane coordinates $\boldsymbol{p}_{v,i}$. If it does not, either the plane pose ${}^{1}\mathrm{T}_{v}$ or the local plane coordinates are incorrect. The occurring error

$$r_{v,i} = |\boldsymbol{p}'_{v,i} - \boldsymbol{p}_{v,i}| \tag{5.1.29}$$

is called the *ray-point distance*. The points $\boldsymbol{p}'_{v,i}$ are the intersections of ray i with plane v as given by the current estimated configuration of rays and planes. All $\boldsymbol{p}'_{v,i}$ are calculated in the following way: first, the viewing ray of pixel i is determined by using the procedure described in the last section. The ray is then transformed to the local coordinate system v by utilizing the inverse of ${}^{1}\mathrm{T}_{v}$. There, it is intersected with the xy-plane, giving the intersection point $\boldsymbol{p}'_{v,i}$. Due to the mentioned errors in the plane poses, $\boldsymbol{p}'_{v,i}$ and $\boldsymbol{p}_{v,i}$ are not the same point.
Minimizing the sum of all $r_{v,i}$ by adjusting the plane poses brings the system closer to the real configuration. Dunne et al. [DMW10] propose to do so by using a two-step approach:

1. Get the ray parameters for pixels $\boldsymbol{p}_{img,i}$ with $i = \{1, ..., N_i\}$ from the plane position measurements and the current values of the plane poses ${}^{1}\mathrm{T}_{v}$.

2. Keep the ray parameters fixed and adjust the ${}^{1}\mathrm{T}_{v}$ such as the summed ray-point distance is minimized. In case of convergence stop the procedure, otherwise go back to step 1.

This method was used to conduct the following experiments for various real imaging systems (also presented in [RW12b]). For comparison, the local plane positions are either determined with active grids by a coded light approach (CLA) or via spline surface interpolation (SS). After determining initial values for the plane poses, the bundle adjustment procedure (BA) is executed in two different ways. Either the optical center of the virtual camera is kept as a point that has to lie on every viewing ray (*central BA*), or the assumption of centrality is abandoned and only the plane intersection points $\boldsymbol{P}_{v,i}$ are used for ray determination (*non-central BA*). The actual optimization in step 2 is done by using an implementation of the Levenberg-Marquardt algorithm. Experiments are conducted for three cameras: a pinhole camera with minor lens distortions, a fisheye camera and a catadioptric device made of a deformed non-planar mirror and a perspective camera. The utilized camera images can be seen in Figures 5.1.6, 5.1.7 and 5.1.8, a summary of the results is

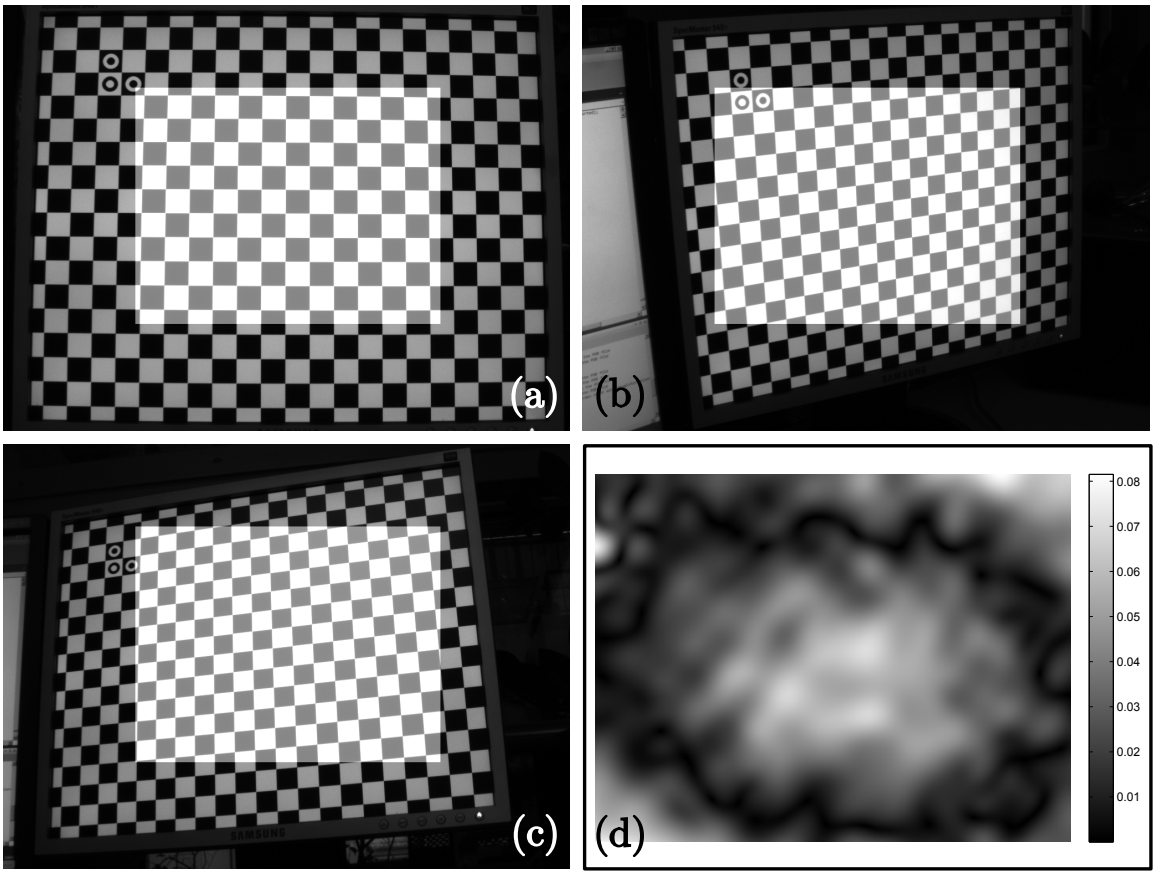


Figure 5.1.6.: Real camera images used for calibrating a pinhole camera. The plane in image (a) is used as virtual image plane, the bright area marks the finally calibrated image region. Images (b) and (c) show the additional views used for calibration. Image (d) displays the spatial distribution of the residual ray-point error in mm after non-central bundle adjustment.

shown in Table 5.1.3. The numbers indicate that the fisheye camera is better described by a non-central model, as the residual ray-point distance decreases when centrality is not enforced. This interpretation is also valid for the catadioptric system and proves that the results obtained under the assumption of a central system are well-suited for initializing the non-central optimization procedure. Another important conclusion drawn from this experiment is that the approach used to gather measurement data hardly influences the quality of the results. This justifies the utilization of spline surfaces with interpolation instead of a coded light approach.
Although these results are promising, the presented approach has two disadvantages:

1. The algorithm converges slowly, especially for non-central cameras.

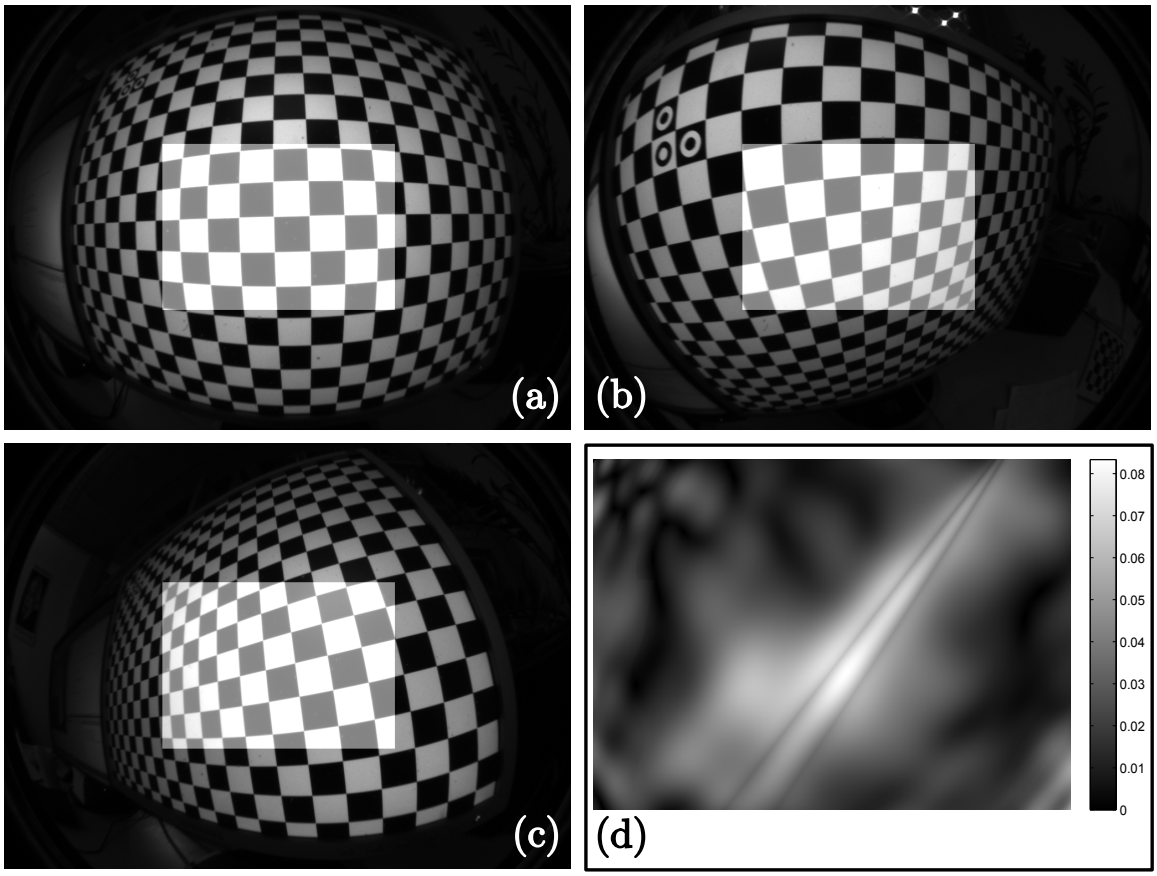


Figure 5.1.7.: Real camera images used for calibrating a camera with a fisheye lens. The plane in image (a) is used as virtual image plane, the bright area marks the finally calibrated image region. Images (b) and (c) show the additional views used for calibration. Image (d) displays the spatial distribution of the residual ray-point error in mm after non-central bundle adjustment.

	before BA	central BA	non-central BA
pinhole + CLA	0.0353 ± 0.0247	0.0340 ± 0.0247	0.0367 ± 0.0350
pinhole + SS	0.0384 ± 0.0279	0.0372 ± 0.0273	0.0336 ± 0.0265
fisheye + CLA	0.1892 ± 0.0985	0.0503 ± 0.0448	0.0354 ± 0.0582
fisheye + SS	0.0410 ± 0.0261	0.0335 ± 0.0243	0.0243 ± 0.0192
catadioptric + CLA	0.3776 ± 0.1569	0.1729 ± 0.1064	0.1353 ± 0.0868
catadioptric + SS	0.5930 ± 0.3264	0.1738 ± 0.1117	0.1440 ± 0.0878

Table 5.1.3.: Results of the calibration procedure in [RW12b]. Values are mean ray-point distances in millimeters, given with their standard deviations (BA = bundle adjustment, CLA = coded light approach, SS = spline surfaces).

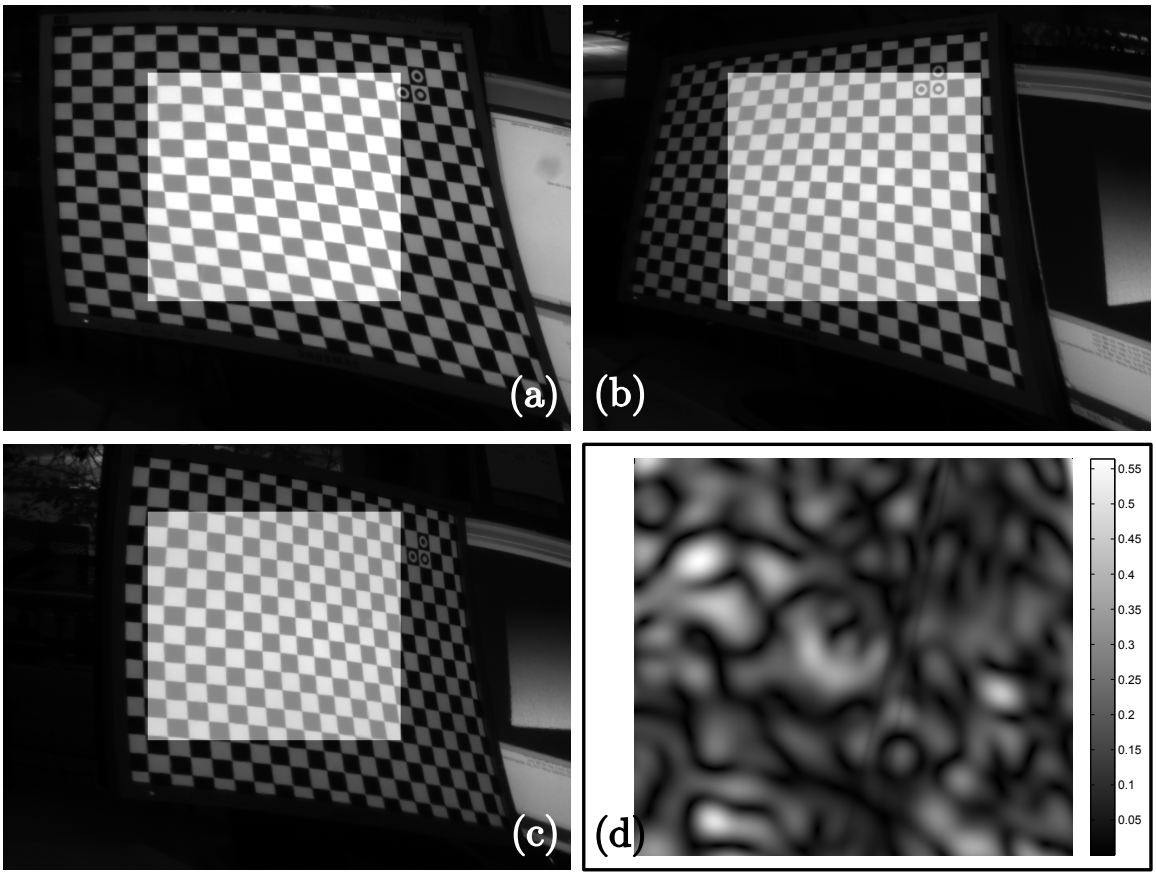


Figure 5.1.8.: Real camera images used for calibrating a catadioptric system with a non-planar bendable mirror and a perspective camera. The plane in image (a) is used as virtual image plane, the bright area marks the finally calibrated image region. Images (b) and (c) show the additional views used for calibration. Image (d) displays the spatial distribution of the residual ray-point error in mm after non-central bundle adjustment.

2. Measurement uncertainties are not used at any point of the procedure. Utilizing them to weigh the input values most likely leads to more accurate results and decreases the influence of measurement errors.

For these reasons, the optimization problem will be reformulated to get a maximum likelihood estimate. To achieve this goal, a mathematical formulation of the ray-point distance is needed that contains ray parameters as well as plane poses.

5.1.6.1. The optimization problem

As mentioned in (2.1.10), the intersection point $\mathbf{C}_{v,i}$ of a line $\mathbf{L}_i = [\boldsymbol{d}_i^T \ \boldsymbol{m}_i^T]^T$ and a plane $\mathbf{V} = [v_1 \ v_2 \ v_3 \ v_h]^T = [\boldsymbol{v}^T \ v_h]^T$ can be determined via

$$\mathbf{C}_{v,i} = \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{m}_i - \boldsymbol{d}_i v_h \\ \boldsymbol{v}^T \boldsymbol{d}_i \end{bmatrix}. \tag{5.1.30}$$

Here, $\mathbf{V}$ is the xy-plane of the coordinate system defined by ${}^1\mathrm{T}_v = \begin{bmatrix} \mathrm{R}_v & \boldsymbol{t}_v \\ \mathbf{0}_3^T & 1 \end{bmatrix}$. The third column $\boldsymbol{r}_{3_v}$ of R_v is a normal vector of that plane which leads to the Hesse normal form

$$\boldsymbol{r}_{3_v}^T(\boldsymbol{x} - \boldsymbol{t}_v) = 0 \tag{5.1.31}$$

$$\Leftrightarrow r_{31_v}x_1 + r_{32_v}x_2 + r_{33_v}x_3 - \boldsymbol{r}_{3_v}^T\boldsymbol{t}_v = 0. \tag{5.1.32}$$

In homogeneous coordinates, the plane representation is now given as

$$\mathbf{V}_v = \begin{bmatrix} \boldsymbol{r}_{3_v} \\ -\boldsymbol{r}_{3_v}^T\boldsymbol{t}_v \end{bmatrix} \tag{5.1.33}$$

which gives the intersection point

$$\mathbf{C}_{v,i} = \begin{bmatrix} \boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i \boldsymbol{r}_{3_v}^T \boldsymbol{t}_v \\ \boldsymbol{r}_{3_v}^T \boldsymbol{d}_i \end{bmatrix}. \tag{5.1.34}$$

As $\mathbf{C}_{v,i}$ is determined in the camera coordinate system, it needs to be transferred to the local plane coordinate system to allow a direct comparison to the local plane position measurement $\boldsymbol{p}_{v,i}$. This can be achieved by multiplication with

$${}^1\mathrm{T}_v^{-1} = \begin{bmatrix} \mathrm{R}_v^T & -\mathrm{R}_v^T\boldsymbol{t}_v \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{5.1.35}$$

leading to

$$\mathbf{P}_{v,i} = {}^1\mathrm{T}_v^{-1}\mathbf{C}_{v,i} = \begin{bmatrix} \mathrm{R}_v^T(\boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i \boldsymbol{r}_{3_v}^T \boldsymbol{t}_v) - \mathrm{R}_v^T \boldsymbol{t}_v \boldsymbol{r}_{3_v}^T \boldsymbol{d}_i \\ \boldsymbol{r}_{3_v}^T \boldsymbol{d}_i \end{bmatrix}. \tag{5.1.36}$$

Euclidean normalization (i.e. division by the last element) and restriction to the first three elements delivers

$$\boldsymbol{P}_{v,i} = \mathrm{R}_v^T(\boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i \boldsymbol{r}_{3_v}^T \boldsymbol{t}_v)\frac{1}{\boldsymbol{r}_{3_v}^T \boldsymbol{d}_i} - \mathrm{R}_v^T \boldsymbol{t}_v. \tag{5.1.37}$$

These points can be used for comparison with the plane position measurements $\boldsymbol{p}_{v,i}^{3D} = (\boldsymbol{p}_{v,i}^T,\ 0)^T$:

$$\boldsymbol{P}_{v,i} \stackrel{!}{=} \boldsymbol{p}_{v,i}^{3D}. \tag{5.1.38}$$

Using this relation, the model function for minimization of the ray-point distances can be defined as:

$$\boldsymbol{f}_{v,i}(\mathrm{R}_v, \boldsymbol{t}_v, \mathbf{L}_i) := \boldsymbol{P}_{v,i} = \boldsymbol{p}_{v,i}^{3D}. \tag{5.1.39}$$

Remark: all points $\boldsymbol{P}_{v,i}$ lie on the local xy-plane and therefore their third elements are always equal to zero. Consequently, each model function has two useful rows and therefore delivers only two independent constraints for the optimization procedure. However, to facilitate the derivation of the Jacobians, the formulation as a 3D vector will be maintained for now.
The main components of this model are:

- the local 2D plane position measurements $\boldsymbol{p}_{v,i}$,
- the line parameters $\mathbf{L}_i = [\boldsymbol{m}_i^T\ \boldsymbol{d}_i^T]^T$,
- the plane poses with rotation matrices R_v and translation vectors $\boldsymbol{t}_v$.

The goal of the optimization procedure is to determine those plane poses that minimize the ray-point distance. This procedure is described in the next section.

5.1.6.2. Minimization of the ray-point distance

The model function is defined as

$$\boldsymbol{f}_{v,i}(\mathrm{R}_v, \boldsymbol{t}_v, \mathbf{L}_i) := \mathrm{R}_v^T(\boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i \boldsymbol{r}_{3_v}^T \boldsymbol{t}_v)\frac{1}{\boldsymbol{r}_{3_v}^T \boldsymbol{d}_i} - \mathrm{R}_v^T \boldsymbol{t}_v = \boldsymbol{p}_{v,i}^{3D}. \tag{5.1.40}$$

In this equation, the local intersections $\boldsymbol{p}_{v,i}^{3D}$ are the measurements. Their covariance matrices can be gained from the uncertain spline surfaces fitted through the chessboard corners as described in Section 5.1.1.1. These matrices serve to build the weight matrix W which is utilized to solve the minimization problem

$$\underset{\mathbf{R_v}, \boldsymbol{t_v}, \mathbf{L_i}}{\operatorname{argmin}}\ \widehat{\boldsymbol{v}}_{v,i}^T \mathrm{W} \widehat{\boldsymbol{v}}_{v,i}. \tag{5.1.41}$$

The $\widehat{\boldsymbol{v}}_{v,i} = \boldsymbol{f}(\widehat{\mathrm{R}}_v, \widehat{\boldsymbol{t}}_v, \widehat{\mathbf{L}}_i) - \boldsymbol{p}_{v,i}^{3D}$ pose as estimated corrections which are the ray-point distances in this case. The procedure of non-linear least squares parameter estimation from Section 2.4.3 is used to get the estimates $\widehat{\mathrm{R}}_v$ and $\widehat{\boldsymbol{t}}_v$, $v = 2..V$. It is assumed that the measurements are random variables with a Gaussian probability distribution, which makes the final solution for the parameters the maximum likelihood estimate.
Instead of the redundant rotation matrix R_v, which needs 9 elements to represent a 3D rotation, the Rodrigues vector $\boldsymbol{R}_v$ (see Section 2.1.4.2 for details) will be used as a parameter. Together with the translation vectors $\boldsymbol{t}_v$ and the Plücker line coordinates $\mathbf{L}_i = (\boldsymbol{d}^T, \boldsymbol{m}^T)^T$ they form the full parameter vector

$$\boldsymbol{x} = [\boldsymbol{R}_2^T, \boldsymbol{t}_2^T, ..., \boldsymbol{R}_V^T, \boldsymbol{t}_V^T, \boldsymbol{d}_1^T, \boldsymbol{m}_1^T, ..., \boldsymbol{d}_{N_i}^T, \boldsymbol{m}_{N_i}^T]^T. \tag{5.1.42}$$

The calibration plane that is used as virtual image plane during initial calibration defines the camera coordinate system. To avoid singularities this plane stays fixed during the bundle adjustment procedure. This means that $\mathrm{R}_1 = \mathrm{I}_3$ and $\boldsymbol{t}_1 = \mathbf{0}$ and that they are not changed.
The local positions on the planes $\mathbf{V}_v$ build the measurement vector

$$\boldsymbol{l} = [\boldsymbol{p}_{1,1}^T, ..., \boldsymbol{p}_{V,N_i}^T]^T. \tag{5.1.43}$$

The most intricate part of the procedure is the determination of the Jacobians. Detailed derivations can be found in Section A.3. With $r_t = \boldsymbol{r}_{3_v}^T \boldsymbol{t}_v$ and $r_d = \boldsymbol{r}_{3_v}^T \boldsymbol{d}_i$ they end up as:

$$\mathrm{J}_{R_v}^{v,i} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{R}_v} = \mathrm{R}_v^T \left(\mathsf{S}(\boldsymbol{d}_i) r_t + \boldsymbol{d}_i \boldsymbol{r}_{3_v}^T \mathsf{S}(\boldsymbol{t}_v) - (\boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i r_t) \frac{\boldsymbol{r}_{3_v}^T \mathsf{S}(\boldsymbol{d}_i)}{r_d} \right) \frac{1}{r_d} \tag{5.1.44}$$

$$- \mathrm{R}_v^T \mathsf{S}(\boldsymbol{t}_v) + \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathrm{R}_v^T \mathsf{S}(\boldsymbol{m}_i) \frac{1}{r_d} \tag{5.1.45}$$

$$\mathrm{J}_{t_v}^{v,i} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{t}_v} = \mathrm{R}_v^T \left(\frac{\boldsymbol{d}_i \boldsymbol{r}_{3_v}^T}{r_d} - \mathrm{I}_3 \right) \tag{5.1.46}$$

$$\mathrm{J}_{d_i}^{v,i} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{d}_i} = \mathrm{R}_v^T \left(\mathrm{I}_3 r_d r_t - (\boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i r_t) \boldsymbol{r}_{3_v}^T \right) \frac{1}{r_d^2} \tag{5.1.47}$$

$$\mathrm{J}_{m_i}^{v,i} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{m}_i} = \mathrm{R}_v^T \mathsf{S}(\boldsymbol{r}_{3_v}) \frac{1}{r_d}. \tag{5.1.48}$$

As mentioned before, only the first two rows of the constraint function deliver valuable information. Therefore, the last row of all Jacobians will be deleted to form the row-reduced matrices $\mathrm{J}_{R_v}^{v,i*}$, $\mathrm{J}_{t_v}^{v,i*}$, $\mathrm{J}_{d_i}^{v,i*}$ and $\mathrm{J}_{m_i}^{v,i*}$.
The used Plücker line coordinates are a redundant homogeneous representation. To avoid further constraints, they will be transferred to a reduced space for optimization. Section 2.4.4 elaborates on this procedure of maximum likelihood estimation with reduced

coordinates. The transformation to reduced coordinates is performed with the help of the projection matrix

$$\mathrm{J_{r,i}} = \mathrm{null}([\mathbf{L}_i\ \overline{\mathbf{L}}_i]^T) \tag{5.1.49}$$

via

$$\mathbf{L}_{r,i} = \mathrm{J_{r,i}}^T \mathbf{L}_i. \tag{5.1.50}$$

This leads to the (2×4) reduced Jacobian for each line intersection:

$$\mathrm{J}^{v,i*}_{L_{r,i}} = [\mathrm{J}^{v,i*}_{d_i}\ \mathrm{J}^{v,i*}_{m_i}]\mathrm{J_{r,i}}. \tag{5.1.51}$$

Now all matrices required to execute the bundle adjustment procedure which minimizes the ray-point distance are available. To start the optimization, initial approximate values $\widehat{\boldsymbol{x}}^a$ of all parameters are needed. They can be gained e.g. from the initial linear calibration procedure described in Section 5.1.3. The plane position measurements $\boldsymbol{p}_{i,v}$ come from inverting the spline surface used for chessboard interpolation of plane v at pixel i. Section 5.1.1.1 shows how the covariance matrix $\Sigma_{u,v}$ of the corresponding spline parameter values can be determined. Scaling with the size of the chessboard delivers the needed covariance $\Sigma_{\boldsymbol{p}_{i,v}\boldsymbol{p}_{i,v}}$. As also described in Section 5.1.2, it is possible to calculate the full covariance matrices $\Sigma_{\boldsymbol{pp}_v}$ of all intersection points on plane v. Due to the intersection points being the result of a spline inversion procedure based on non-independent spline control points, the $\Sigma_{\boldsymbol{pp}_v}$ might become singular. A possible workaround to avoid this problem is given in the list of requirements for a successful bundle adjustment procedure later on in this section. Line measurements and covariances are gained by the procedure described in Section 5.1.5.

One model function $\boldsymbol{f}_{v,i}(\widehat{\boldsymbol{x}})$ is given for each intersection of a viewing ray i with a calibration plane v. Each of them delivers two equations. Stacking them leads to the design matrix

$$\mathrm{A} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\widehat{\boldsymbol{x}}^a} = \left[\begin{array}{cccccccc} \mathrm{J}^{1,1*}_{R_2} & \mathrm{J}^{1,1*}_{t_2} & \dots & \mathrm{J}^{1,1*}_{R_V} & \mathrm{J}^{1,1*}_{t_V} & \mathrm{J}^{1,1*}_{L_{r,1}} & \dots & \mathrm{J}^{1,1*}_{L_{r,N_i}} \\ \mathrm{J}^{2,1*}_{R_2} & \mathrm{J}^{2,1*}_{t_2} & \dots & \mathrm{J}^{2,1*}_{R_V} & \mathrm{J}^{2,1*}_{t_V} & \mathrm{J}^{2,1*}_{L_{r,1}} & \dots & \mathrm{J}^{2,1*}_{L_{r,N_i}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{J}^{V,N_i*}_{R_2} & \mathrm{J}^{V,N_i*}_{t_2} & \dots & \mathrm{J}^{V,N_i*}_{R_V} & \mathrm{J}^{V,N_i*}_{t_V} & \mathrm{J}^{V,N_i*}_{L_{r,1}} & \dots & \mathrm{J}^{V,N_i*}_{L_{r,N_i}} \end{array}\right]_{\boldsymbol{x}=\widehat{\boldsymbol{x}}^a}. \tag{5.1.52}$$

The measurement covariance matrix reads as

$$\Sigma_{ll} = \left[\begin{array}{ccc} \Sigma_{\boldsymbol{pp}_1} & & 0 \\ & \ddots & \\ 0 & & \Sigma_{\boldsymbol{pp}_V} \end{array}\right]. \tag{5.1.53}$$

Updating the parameters is done as depicted in Section 2.4.3:

$$\widehat{\Delta \boldsymbol{x}} = (\mathrm{A}^T \Sigma_{ll}^{-1} \mathrm{A})^{-1} \mathrm{A}^T \Sigma_{ll}^{-1} \Delta \boldsymbol{l}. \tag{5.1.54}$$

For the following descriptions of the updating processes, the subscripts of the update vector $\widehat{\Delta \boldsymbol{x}}$ indicate, which elements are used.

- Plane rotation parameters:

$$\widehat{\mathrm{R}}^a_{v,\text{updated}} = \widehat{\mathrm{R}}^a_v \cdot \mathrm{R}(\widehat{\Delta \boldsymbol{R}}^a_v) \quad \text{with} \quad \widehat{\Delta \boldsymbol{R}}^a_v = \widehat{\Delta \boldsymbol{x}}\Big|_{p_{R_v}..p_{R_v}+2} \tag{5.1.55}$$

  Here, $p_{R_v} = 6(v-2)+1$ stands for the position of Rodrigues vector $\widehat{\boldsymbol{R}}^a_v$ in the parameter vector ($v = 2..V$). A concatenation of rotations can be done via multiplication of the corresponding rotation matrices. Therefore, an incremental Rodrigues update vector $\widehat{\Delta \boldsymbol{R}}^a_v$ first needs to be converted to a rotation matrix, e.g. by using the Rodrigues formula (2.1.23). Then, the updated rotation matrix $\widehat{\mathrm{R}}^a_{v,\text{updated}}$ can be determined as shown in (5.1.55).

- Plane translation parameters:

$$\widehat{\boldsymbol{t}}^a_{v,\text{updated}} = \widehat{\boldsymbol{t}}^a_v + \widehat{\Delta \boldsymbol{x}}\Big|_{p_{R_v}+3..p_{R_v}+5} \tag{5.1.56}$$

- Line parameters:

$$\widehat{\mathbf{L}}^{a'}_{i,\text{updated}} = \widehat{\mathbf{L}}^a_i + \mathrm{J}_{r,i}\widehat{\Delta \boldsymbol{x}}\Big|_{p_{L_i}..p_{L_i}+3} \tag{5.1.57}$$

  Here, $p_{L_i} = 6(V-1)+1+4(i-1)$ is the position of the elements of line i in the update vector. As the differential updates are determined in a reduced space, they need to be converted to the original Plücker space to perform the update. The resulting 6D vector $\widehat{\mathbf{L}}^{a'}_{i,\text{updated}}$ is not necessarily a valid Plücker coordinate. By enforcing the Plücker constraint as explained in Section 2.1.3, this can be remedied. Subsequent spherical normalization delivers the new line parameter estimate $\widehat{\mathbf{L}}^a_i$.

In contrast to the method used in [RW12b], this is a one-step procedure as plane poses and line parameters are updated at the same time. Furthermore, the viewing rays are only defined by their intersections with the calibration planes and not by a fixed optical center. This allows to converge towards a non-central solution.
The iterative procedure of determining the updates and actual updating is repeated until convergence, e.g. until $|\widehat{\Delta \boldsymbol{x}}|$ drops below a specified threshold.
In a last step, the covariance matrix of the final parameter estimate is determined as

$$\Sigma_{\widehat{\boldsymbol{x}}^a\widehat{\boldsymbol{x}}^a} = (\mathrm{A}^T\Sigma^{-1}_{\boldsymbol{ll}}\mathrm{A})^{-1}. \tag{5.1.58}$$

The (3×3) covariance matrices $\Sigma_{\boldsymbol{R}_v\boldsymbol{R}_v}$ and $\Sigma_{\boldsymbol{t}_v\boldsymbol{t}_v}$ of the Rodrigues and translation vectors $\boldsymbol{R}_v$ and $\boldsymbol{t}_v$ can be found along the diagonal of $\Sigma_{\widehat{\boldsymbol{x}}^a\widehat{\boldsymbol{x}}^a}$. The uncertainties of all involved lines, described by $\Sigma_{\mathbf{LL}}$, are also available, but will not be used in the following. Also all

other off-diagonal entries of $\Sigma_{\widehat{\boldsymbol{x}}^a\widehat{\boldsymbol{x}}^a}$ are disregarded from now on.

$$\Sigma_{\widehat{\boldsymbol{x}}^a\widehat{\boldsymbol{x}}^a} = \begin{bmatrix} \Sigma_{\boldsymbol{R}_2\boldsymbol{R}_2} & & & & & & \\ & \Sigma_{\boldsymbol{t}_2\boldsymbol{t}_2} & & & & & \\ & & \ddots & & & & \\ & & & \Sigma_{\boldsymbol{R}_V\boldsymbol{R}_V} & & & \\ & & & & \Sigma_{\boldsymbol{t}_V\boldsymbol{t}_V} & & \\ & & & & & \Sigma_{\mathbf{LL}} \end{bmatrix}. \tag{5.1.59}$$

For the bundle adjustment procedure to succeed, a few requirements have to be met:

- Each calibration plane has to be intersected by at least 3 viewing rays. This delivers the 6 constraints that are necessary to avoid an underdetermined system of equations.

- Every viewing ray has to intersect more than 2 calibration planes. Otherwise it would be a perfect fit, regardless of the correctness of the plane poses. The corresponding ray-point distances were exactly 0, leaving no error to be minimized.

- Singular measurement covariance matrices $\Sigma_{\boldsymbol{p}\boldsymbol{p}_v}$ have to be avoided. A reason for singularity can be that the plane intersection points lie close together. In that case, the generated interpolated points are all based on the same input data, i.e. the same chessboard corners. This eventually leads to an increased condition number of the covariance matrix. If plane points in close proximity cannot be avoided, setting the entries of the covariance matrix that describe correlations between them to 0 can help to prevent singularity.

- One of the involved calibration planes has to remain static. This roughly fixes the position of the planes in the camera coordinate system and prevents drifting. This requirement is already met by the equations formulated above, as the intersection points $\boldsymbol{p}_{1,i}$ of the viewing rays with the virtual image plane are part of the measurement vector, but the corresponding plane pose vectors $\boldsymbol{R}_1$ and $\boldsymbol{t}_1$ are no members of the parameter vector.

- In case of a central camera one degree of freedom remains as there are no further fixations in the system than the constant intersections of the viewing rays with the virtual image plane. It is therefore possible to change the distance of the calibration planes from the caustic center with an according adjustment of the viewing ray directions. This phenomenon is similar to the issue of the unknown scale factor in the classical *structure from motion* problem (see [HZ03]). There are various possibilities to stabilize the procedure, e.g. fixing a second plane or keeping the average position of the planes constant. Unfortunately, they are too restrictive to allow convergence

towards the best solutions in case the current system state is far from it. Therefore, a simple practical solution is proposed here: the closeness of the current system state to the best solutions can be assessed by determining the ray-point distances. If they are higher than to be expected, the optimization procedure is executed without making use of the additional constraint. This brings the system quickly close to the best solutions. Although it theoretically becomes unstable at that point, practical experiments showed that the procedure does not diverge. This is caused by the fact that the system data is always influenced by measurement errors and therefore the singularity is never reached exactly. However, to prevent sliding and avoid instability, the current position $\boldsymbol{t}_f$ of a selected plane f will be used as a measurement in case the system state is close to the solution space. This prevents the system from moving from the current position within the solution space. It is realized by adding the additional constraint

$$\boldsymbol{f}_f := \boldsymbol{t}_f = \boldsymbol{t}_f^a. \tag{5.1.60}$$

The current approximation $\boldsymbol{t}_f^a$ is attached to the measurement vector, its covariance matrix augments the overall covariance matrix Σ_{ll} and the identity matrix $\mathrm{I}_3 = \frac{\partial \boldsymbol{f}_f}{\partial \boldsymbol{t}_f}$ is added at the corresponding position of the design matrix from (5.1.52):

$$\mathrm{A}_f = \begin{bmatrix} & & \mathrm{A} & & \\ 0_{3,2} & \dots & 0_{3,f-1} & \mathrm{I}_3 & \dots \end{bmatrix}.$$

In practice, the complete procedure is best executed in the following way:

1. Evaluate the closeness of the system to the solution space by determining the ray-point distances for all calibration planes.

2. If the ray-point distance of one of the planes is comparably high: conclude that the system is far from a correct solution and execute optimization without the additional constraint (5.1.60).

3. Execute optimization with additional constraint.

5.1.6.3. Solution refinement

The procedure described in the last section delivers good results. However, the accuracy of the utilized viewing rays can be very low for certain configurations of the calibration planes. Especially, when the planes lie near by each other or even intersect, the ray parameters can be inaccurate as the points used for fitting lie close together. To handle this problem, this section provides another optimization method that considers the line

parameters also as measurements. This allows their covariance matrices to be used to form weight matrices and therefore decrease the influence of line parameters with reduced accuracy. First, the model is converted to an implicit form:

$$\boldsymbol{g}_{v,i}(\mathrm{R}_v, \boldsymbol{t}_v, \mathbf{L}_i, \boldsymbol{p}_{v,i}) = \boldsymbol{P}_{v,i} - \boldsymbol{p}_{v,i}^{3D} = \mathbf{0}_3. \tag{5.1.61}$$

Now, only the plane poses, represented by Rodrigues and translation vectors, form the parameter vector

$$\boldsymbol{x} = [\boldsymbol{R}_2^T, \boldsymbol{t}_2^T, ..., \boldsymbol{R}_V^T, \boldsymbol{t}_V^T]^T. \tag{5.1.62}$$

The virtual image plane is still fixed during the optimization procedure, therefore $\mathrm{R}_1 = \mathrm{I}_3$ and $\boldsymbol{t}_1 = \mathbf{0}_3$. All remaining components are considered as measurements and build the measurement vector

$$\boldsymbol{l} = [\boldsymbol{p}_{1,1}^T, ..., \boldsymbol{p}_{V,N_i}^T, \mathbf{d}_1^T, \mathbf{m}_1^T, ..., \mathbf{d}_{N_i}^T, \mathbf{m}_{N_i}^T]^T. \tag{5.1.63}$$

The Jacobians are the same as before for the most part, only joined by the partial derivatives of the plane intersection points:

$$\mathrm{J}_{R_v}^{v,i} = \frac{\partial \boldsymbol{g}_{v,i}}{\partial \boldsymbol{R}_v} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{R}_v}, \quad \mathrm{J}_{t_v}^{v,i} = \frac{\partial \boldsymbol{g}_{v,i}}{\partial \boldsymbol{t}_v} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{t}_v} \tag{5.1.64}$$

$$\mathrm{J}_{d_i}^{v,i} = \frac{\partial \boldsymbol{g}_{v,i}}{\partial \boldsymbol{d}_i} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{d}_i}, \quad \mathrm{J}_{m_i}^{v,i} = \frac{\partial \boldsymbol{g}_{v,i}}{\partial \boldsymbol{m}_i} = \frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{m}_i} \tag{5.1.65}$$

$$\mathrm{J}_{p_{v,i}}^{v,i} = \frac{\partial \boldsymbol{g}_{v,i}}{\partial \boldsymbol{p}_{v,i}} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix}. \tag{5.1.66}$$

Still, only the first two rows of the constraint function are used, leading to the row-reduced matrices $\mathrm{J}_{R_v}^{v,i*}$, $\mathrm{J}_{t_v}^{v,i*}$, $\mathrm{J}_{p_{v,i}}^{v,i*}$, $\mathrm{J}_{d_i}^{v,i*}$ and $\mathrm{J}_{m_i}^{v,i*}$.
The Plücker line coordinates are a redundant homogeneous representation which has a singular covariance matrix $\Sigma_{\mathbf{L}_i\mathbf{L}_i}$. A further benefit of working in reduced spaces is that the covariance matrices become regular, allowing their inverses to be used as weight matrices during the optimization procedure. Projection of the $\mathbf{L}_i$ to the tangent space of the unit sphere (for further details, see Section 2.4.4) is realized via

$$\mathbf{L}_{r,i} = {\mathrm{J}_{\mathrm{r,i}}}^T \mathbf{L}_i \quad \text{with} \quad \mathrm{J}_{\mathrm{r,i}} = \mathrm{null}([\mathbf{L}_i\ \overline{\mathbf{L}}_i]^T). \tag{5.1.67}$$

The corresponding non-singular covariance matrix in reduced space is calculated as

$$\Sigma_{\mathbf{L}_{r,i}\mathbf{L}_{r,i}} = {\mathrm{J}_{\mathrm{r,i}}}^T \Sigma_{\mathbf{L}_i\mathbf{L}_i} \mathrm{J}_{\mathrm{r,i}}. \tag{5.1.68}$$

This gives the (2×4) reduced Jacobian for each line intersection

$$\mathrm{J}_{L_{r,i}}^{v,i*} = [\mathrm{J}_{d_i}^{v,i*}\ \mathrm{J}_{m_i}^{v,i*}]\mathrm{J}_{\mathrm{r,i}}. \tag{5.1.69}$$

To start the non-linear optimization procedure, initial approximate values $\widehat{\boldsymbol{x}}^a$ of all parameters and $\widehat{\boldsymbol{l}}^a$ of all measurements, as well as the corresponding covariance matrices are needed. The parameters are known either from the initial calibration procedure or as the result of the optimization method from the last section. Plane position measurements $\boldsymbol{p}_{v,i}$ and their complete covariance matrices for each plane $\Sigma_{\boldsymbol{pp}_v}$ are determined as before.
Stacking the first two rows $\boldsymbol{g}^*_{v,i}(\widehat{\boldsymbol{l}}, \widehat{\boldsymbol{x}})$ of the model functions forms the equation system of the constraints

$$\boldsymbol{g}(\widehat{\boldsymbol{l}}, \widehat{\boldsymbol{x}}) = \boldsymbol{g}(\widehat{\boldsymbol{l}}^a, \widehat{\boldsymbol{x}}^a) + \mathrm{A}\widehat{\Delta\boldsymbol{x}} + \mathrm{B}^T\widehat{\Delta\boldsymbol{l}} = \boldsymbol{0}. \tag{5.1.70}$$

This leads to the optimization matrices:

$$\mathrm{A} = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\widehat{\boldsymbol{x}}^a \\ \boldsymbol{l}=\widehat{\boldsymbol{l}}^a}} = \begin{bmatrix} \mathrm{J}^{1,1*}_{R_2} & \mathrm{J}^{1,1*}_{t_2} & \dots & \mathrm{J}^{1,1*}_{R_V} & \mathrm{J}^{1,1*}_{t_V} \\ \mathrm{J}^{2,1*}_{R_2} & \mathrm{J}^{2,1*}_{t_2} & \dots & \mathrm{J}^{2,1*}_{R_V} & \mathrm{J}^{2,1*}_{t_V} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{J}^{V,N_i*}_{R_2} & \mathrm{J}^{V,N_i*}_{t_2} & \dots & \mathrm{J}^{V,N_i*}_{R_V} & \mathrm{J}^{V,N_i*}_{t_V} \end{bmatrix}_{\substack{\boldsymbol{x}=\widehat{\boldsymbol{x}}^a \\ \boldsymbol{l}=\widehat{\boldsymbol{l}}^a}} \tag{5.1.71}$$

$$\mathrm{B}^T = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{l}}\right|_{\substack{\boldsymbol{x}=\widehat{\boldsymbol{x}}^a \\ \boldsymbol{l}=\widehat{\boldsymbol{l}}^a}} = \begin{bmatrix} \mathrm{J}^{1,1*}_{p_{1,1}} & \dots & \mathrm{J}^{1,1*}_{p_{V,N_i}} & \mathrm{J}^{1,1*}_{L_{r,1}} & \dots & \mathrm{J}^{1,1*}_{L_{r,N_i}} \\ \mathrm{J}^{2,1*}_{p_{2,1}} & \dots & \mathrm{J}^{2,1*}_{p_{V,N_i}} & \mathrm{J}^{2,1*}_{L_{r,1}} & \dots & \mathrm{J}^{2,1*}_{L_{r,N_i}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{J}^{V,N_i*}_{p_{1,1}} & \dots & \mathrm{J}^{V,N_i*}_{p_{V,N_i}} & \mathrm{J}^{V,N_i*}_{L_{r,1}} & \dots & \mathrm{J}^{V,N_i*}_{L_{r,N_i}} \end{bmatrix}_{\substack{\boldsymbol{x}=\widehat{\boldsymbol{x}}^a \\ \boldsymbol{l}=\widehat{\boldsymbol{l}}^a}} \tag{5.1.72}$$

$$\widehat{\Sigma}_{\boldsymbol{l}^a\boldsymbol{l}^a} = \begin{bmatrix} \Sigma_{\boldsymbol{pp}_1} & & & & & \\ & \ddots & & & 0 & \\ & & \Sigma_{\boldsymbol{pp}_V} & & & \\ & & & \widehat{\Sigma}_{\mathbf{L}^a_{r,1}\mathbf{L}^a_{r,1}} & & \\ & 0 & & & \ddots & \\ & & & & & \widehat{\Sigma}_{\mathbf{L}^a_{r,N_i}\mathbf{L}^a_{r,N_i}} \end{bmatrix}. \tag{5.1.73}$$

The parameter and measurement updates are determined as shown in Section 2.4.3:

$$\widehat{\Delta\boldsymbol{x}} = (\mathrm{A}^T(\mathrm{B}^T\Sigma_{\boldsymbol{l}^a\boldsymbol{l}^a}\mathrm{B})^{-1}\mathrm{A})^{-1}\mathrm{A}^T(\mathrm{B}^T\Sigma_{\boldsymbol{l}^a\boldsymbol{l}^a}\mathrm{B})^{-1}\boldsymbol{c}_g \tag{5.1.74}$$

$$\widehat{\Delta\boldsymbol{l}} = \Sigma_{\boldsymbol{l}^a\boldsymbol{l}^a}\mathrm{B}(\mathrm{B}^T\Sigma_{\boldsymbol{l}^a\boldsymbol{l}^a}\mathrm{B})^{-1}(\boldsymbol{c}_g - \mathrm{A}\widehat{\Delta\boldsymbol{x}}) - (\widehat{\boldsymbol{l}}^a - \boldsymbol{l}) \tag{5.1.75}$$

with

$$\boldsymbol{c}_g = -\boldsymbol{g}(\widehat{\boldsymbol{l}}^a, \widehat{\boldsymbol{x}}^a) + \mathrm{B}^T(\widehat{\boldsymbol{l}}^a - \boldsymbol{l}). \tag{5.1.76}$$

The updates of the Rodrigues rotation vectors $\widehat{\mathrm{R}}^a_{v,\mathrm{updated}}$ and the translation vectors $\widehat{\boldsymbol{t}}^a_{v,\mathrm{updated}}$ stay the same as before. Now, also the measurements have to be updated:

- Ray-plane intersection measurements:

$$\begin{bmatrix} \widehat{\boldsymbol{p}}^a_{1,1,\mathrm{updated}} \\ \vdots \\ \widehat{\boldsymbol{p}}^a_{V,N_i,\mathrm{updated}} \end{bmatrix} = \left.\widehat{\boldsymbol{l}}^a\right|_{1..2n_p} + \left.\widehat{\Delta\boldsymbol{l}}\right|_{1..2n_p} \tag{5.1.77}$$

n_p is the overall number of ray-plane intersections. Updating the plane intersection measurements $\widehat{\boldsymbol{p}}^a_{v,i}$ is done by a summation, which corresponds to shifting them within the plane. Consequently, their covariance matrices $\Sigma_{\boldsymbol{p}_{v,i}\boldsymbol{p}_{v,i}}$ do not need to be adjusted.

- Line measurements:
$$\widehat{\mathbf{L}}^{a'}_{i,\text{updated}} = \widehat{\mathbf{L}}^a_i + \mathrm{J}_{r,i}\widehat{\Delta\boldsymbol{l}}\Big|_{p_{L_i}..p_{L_i}+3} \tag{5.1.78}$$
In this case, $p_{L_i} = 2n_p + 1 + 4i$ is the position of the elements of line i in the measurement update vector. Conversion back to the original Plücker space to get the new line measurement estimate $\widehat{\mathbf{L}}^a_i$ is done as before by enforcing the Plücker constraint and subsequent spherical normalization.
For each line measurement, also the corresponding covariance matrix $\widehat{\Sigma}_{\mathbf{L}^a_i\mathbf{L}^a_i}$ has to be adjusted. This can be done by determining the rotation matrix $\mathrm{R}_{\mathbf{L}^a_i\widehat{\mathbf{L}}^a_i}$ that maps the original line coordinates $\mathbf{L}^a_i$ to the current approximations $\widehat{\mathbf{L}}^a_i$ via $\widehat{\mathbf{L}}^a_i = \mathrm{R}_{\mathbf{L}^a_i\widehat{\mathbf{L}}^a_i}\mathbf{L}^a_i$ (see Section 2.1.4.3). The updated covariance matrix is then delivered by linear uncertainty propagation:
$$\widehat{\Sigma}_{\mathbf{L}^a_i\mathbf{L}^a_i,updated} = \mathrm{R}_{\mathbf{L}^a_i\widehat{\mathbf{L}}^a_i}\Sigma_{\mathbf{L}^a_i\mathbf{L}^a_i}\mathrm{R}^T_{\mathbf{L}^a_i\widehat{\mathbf{L}}^a_i}. \tag{5.1.79}$$

These optimization steps are repeated until convergence. The final parameter covariance matrix is calculated via
$$\Sigma_{\widehat{\boldsymbol{x}}^a\widehat{\boldsymbol{x}}^a} = (\mathrm{A}^T(\mathrm{B}^T\widehat{\Sigma}_{\boldsymbol{l}^a\boldsymbol{l}^a}\mathrm{B})^{-1}\mathrm{A})^{-1}. \tag{5.1.80}$$
which provides the covariance matrices $\Sigma_{\boldsymbol{R}_v\boldsymbol{R}_v}$ and $\Sigma_{\boldsymbol{t}_v\boldsymbol{t}_v}$ of the Rodrigues and translation vectors $\boldsymbol{R}_v$ and $\boldsymbol{t}_v$ as before.

5.1.6.4. Evaluation of the bundle adjustment procedure

To test the bundle adjustment procedures proposed in the previous sections, again a simulated camera with a fisheye lens and a non-central catadioptric system with a conic mirror are analyzed. For each single experiment, a virtual image plane from a fixed set of calibration planes is chosen. Then, overlapping views are selected to determine initial approximate values $\widehat{\boldsymbol{R}}^a_v$ and $\widehat{\boldsymbol{t}}^a_v$ for plane rotations and translations (see Section 5.1.3). Subsequently, the result is refined by minimizing the ray-point distance with the procedure described in Section 5.1.6.2. The virtual image plane is always the static one, its pose covariances are set to small values to avoid singular matrices during uncertain line fitting (see the remark in Section 5.1.5). Pixel positions $\boldsymbol{p}_{img,i}$ are selected such as at least 3 calibration planes are seen. Furthermore, the $\boldsymbol{p}_{img,i}$ should not lie in close proximity to each other in the camera image. The pixel positions lead to the local intersections $\boldsymbol{p}_{v,i}$ of

line i and plane v. Estimates for the plane uncertainties needed to determine uncertain line points are calculated as shown in Section 5.1.4. Initial values for the line parameters $\widehat{\mathbf{L}}_i^a$ are then determined via maximum likelihood estimation (Section 5.1.5).
In this case, there are $6(V-1)$ pose parameters and $4N_i$ reduced line parameters to be optimized, which requires at least $(6(V-1)+4N_i)/2$ ray-plane intersections to avoid an underdetermined equation system. It is sought to get $1.5 \cdot 4N_i + 4 \cdot 6(V-1)$ equations, requiring each ray to intersect at least 3 planes (giving 6 equations per ray) and a redundancy of 4 for the plane poses.
Depending on the actual overlap of the calibration planes in the image, it is sometimes necessary to pick viewing rays for pixels that lie close together. This invokes the problem of singular plane intersection covariance matrices $\Sigma_{\boldsymbol{pp}_v}$, preventing to use their inverses as weight matrices, but can be remedied by setting the covariances between different points to zero. The update process is continued until $|\widehat{\Delta \boldsymbol{x}}|$ drops below a value of $1 \cdot 10^{-6}$.
There are various issues which can be the cause for a failing calibration procedure. Sometimes they can not be avoided, especially as the simulated experiments are conducted with an automatically generated set of calibration planes and randomly selected pixel positions for ray determination. The main problems are:

- Failing linear calibration.
  Reasons: singular configurations of calibration planes (e.g. parallel planes for a central system) or inappropriate assumption of centrality for non-central systems.
  Solutions: select a different virtual image plane or a completely different set of calibration planes.

- Non-converging bundle adjustment procedure.
  Reasons: results for the initial plane poses are too far from the ground truth or the randomly picked rays are in a singular configuration (e.g. all lying close to a common plane). Also the condition of the covariance matrix $\Sigma_{\boldsymbol{pp}_v}$ used to form weight matrices might be too high.
  Solutions: select a different set of rays and make sure that the covariance matrix $\Sigma_{\boldsymbol{pp}_v}$ is not close to singular.

- Greatly increased uncertainty of the plane positions after bundle adjustment.
  Reasons: some of the viewing rays are highly uncertain, which can e.g. be a consequence of line points that lie close together due to intersecting planes.
  Solutions: ignore viewing rays with high uncertainties for the optimization procedure.

If one of these issues is detected, the procedure is said to be unsuccessful and the results will be ignored. A success rate is calculated as

$$s = 1 - \frac{\text{number of failed procedures}}{\text{number of all experiments}}$$

to indicate the stability of the method.
For evaluation, a simulation is used and therefore the ground truth values of the plane positions $\boldsymbol{t}_{GT,v}$ and the roll, pitch and yaw angles $\Phi_{GT,v}$, $\Theta_{GT,v}$ and $\Psi_{GT,v}$ are known. The following values are determined to evaluate the final results:

- $\overline{r} = \frac{1}{N_{rp}} \sum_{v,i} |\boldsymbol{p}'_{v,i} - \boldsymbol{p}_{v,i}|$ is the average ray-point distance (confer Section 5.1.6) of the current configuration. N_{rp} stands for the number of all ray plane intersections. In case an optimization procedure was executed, $\overline{r}$ is determined for the viewing rays that contributed. Otherwise, rays are randomly picked from the currently calibrated area.

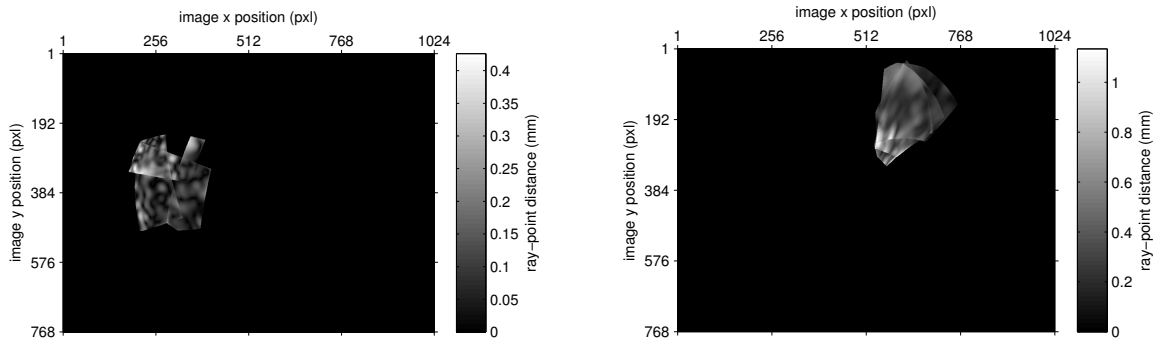
- $\overline{d}_\alpha = \frac{1}{V-1} \sum_{v=2}^{V} (|\Phi_v - \Phi_{GT,v}| + |\Theta_v - \Theta_{GT,v}| + |\Psi_v - \Psi_{GT,v}|)$ gives the mean angular error for all planes.

- $\overline{d}_t = \frac{1}{V-1} \sum_{v=2}^{V} |\boldsymbol{t_v} - \boldsymbol{t}_{GT,v}|$ represents the mean distance of the calibration plane from its ground truth value.

- $\overline{d}_\% = \frac{1}{V-1} \sum_{v=2}^{V} \frac{|\boldsymbol{t_v} - \boldsymbol{t}_{GT,v}|}{|\boldsymbol{C}_{caustic} - \boldsymbol{t}_{GT,v}|}$ is the ratio of the positional error and the actual distance of the calibration plane from the caustic center $\boldsymbol{C}_{caustic}$ of the simulated camera. $\boldsymbol{C}_{caustic}$ is the focal point for the fisheye camera and a point on the rotational axis of the mirror for the conic catadioptric camera. This value expresses the relative accuracy of the distance error and makes it easier to compare the results of different configurations.

- $\overline{T}_{\boldsymbol{tt}} = \frac{1}{V-1} \sum_{v=2}^{V} Tr(\Sigma_{\boldsymbol{t}_v \boldsymbol{t}_v})$ is the mean trace of the covariance matrices $\Sigma_{\boldsymbol{t}_v \boldsymbol{t}_v}$ of the plane translations.

For the experiments, $V = 5$ planes are used. One serves as virtual image plane and therefore defines the camera coordinate system. The relative poses ${}^1\mathrm{T}_v$ of the other planes are determined and compared to the ground truth. The only measurements are the chessboard corners in the camera images. They are not determined by a computer vision procedure in this case but given exactly by the simulation. This allows to perturb them with a defined amount of noise and analyze the effect on the whole calibration procedure. The noise vectors are generated by an isotropic normal distribution with a standard deviation of σ_p and the corresponding covariance matrix $\mathrm{I}_2 \sigma_p^2$.
Tables 5.1.4 and 5.1.5 show the calibration results of the simulated fisheye and conic catadioptric camera, respectively. Exemplary distributions of the residual ray-point distance



(a) (b)

Figure 5.1.9.: Residual ray-point distances after the initial calibration procedure including bundle adjustment and refinement was executed. In each case, 5 calibration planes are used. The results for the simulated fisheye camera are shown in (a), those for the simulated catadioptric system can be seen in (b).

for both cameras can be seen in Figure 5.1.9.
Before elaborating on the specific values, a short comparison with the results of the procedure from [RW12a] (see Table 5.1.3) will be given. The following final numbers for the ray-point distance are higher than in [RW12a], although the calibration procedure is much more advanced and more planes are used. This is a consequence of analyzing multiple configurations and calculating the mean results here. Before, the used configuration was specifically designed to avoid the mentioned problems and therefore deliver good results. Furthermore, the residual error can be higher even though the final result is closer to the ground truth. The reason is that outliers are not eliminated at all costs. So they are still present after the optimization and contribute to an increased average ray-point distance, although the result is actually better than before.
For the current experiments, 41 different sets of calibration planes were analyzed for the fisheye camera and 20 for the catadioptric system. When regarding the final results it has to be kept in mind that all errors are mean values. Because the image positions used for ray determination are selected randomly, certain variations occur during the evaluation procedure. This can lead to results that may seem inconsistent at first sight, e.g. that for the catadioptric system the residual ray-point distance $\overline{r}$ after the initial calibration procedure is highest for the lowest noise level σ_p. The most important result, however, is that the mean ray-point distance is in general decreased by the bundle adjustment and also the refinement procedure. This shows that the optimization method is generally stable, if not one of the basic problems described above occurred. The success rate of the complete process is around 90% for the fisheye camera (see Table 5.1.4). For the conic camera it lies between 50% and 60% (compare Table 5.1.5), the main reason for the decreased rate being that for this non-central system the first step of linear calibration is more likely to fail. For both cameras, increased pixel noise σ_p also leads to higher error values. The

results for the very low noise level of $\sigma_p = 10^{-7}$ pixels gives an idea about how accurate the procedure could get in case the chessboard corner positions were almost perfectly known. For both cameras the final mean relative position error $\overline{d}_{\%}$ lies well below 1%, the angular error is smaller than 1/4th of a degree. With a noise level of $\sigma_p = 0.1$ pixels, which can be considered as realistic for the case that the chessboard corners are actually extracted from the image by a computer vision procedure, the relative translation error is still around 1% for both cameras, the angular error less than 0.5 degrees. For a corner noise of 0.5 pixels, the error values are already much higher. Then, the relative translation error reaches 3.5% for the fisheye and almost 7% for the catadioptric camera. The angular errors lie at 1.2 and 2.6 degrees, respectively.
The traces of the covariance matrices $\overline{T}_{\boldsymbol{tt}}$ give an impression of how the estimated uncertainties change along the calibration procedure. As could be expected, they decrease with each step. For the catadioptric system the uncertainties are higher than for the fisheye camera after the initial calibration, but reach approximately the same values after the last step.
A striking fact is that for the catadioptric camera, the errors are rather big after the initial calibration. This is a consequence of the procedure assuming the camera system to be central, which is not the case for this setup. Nevertheless, the results serve as initial values for the bundle adjustment procedure which dramatically reduces the values of ray-point distance, translational and rotational error. The final error values are close to the ones for the fisheye camera, which proves that the exact same calibration procedure works for central and non-central camera systems.
This concludes the section on the initial calibration procedure. In the following section it will be shown how, based on these results, a calibration of the complete camera image can be obtained.

5.2. Complete calibration

So far, only a small region of the image is calibrated with the initial calibration procedure described in the last section. How this area can be expanded is shown in [RW12a] and illustrated in Figure 3.3.3. If the area which is covered by an uncalibrated view overlaps with the already calibrated image region, known viewing rays from within the common area can be used to formulate an equation system which is linear in the pose parameters of the new plane. In contrast to other works, the proposed approach is neither a minimal one with multiple solutions (as the one of Ramalingam in [RSL05]) nor does it rely on the camera system to be central (as does the proposition of Dunne et al. in [DMW10]). Using this procedure to initialize new planes and afterwards executing the bundle adjustment which minimizes the ray-point distance iteratively expands the calibrated image region. After adding more and more views, finally the complete image is calibrated. In [RW12a] the

			before BA	after BA	after refinement
$\sigma_p = 10^{-7}$pxl	$s = 90.24\%$	$\overline{r}$	0.0326mm	0.0200mm	0.0168mm
		$\overline{d}_\alpha$	0.0525°	0.0782°	0.0758°
		$\overline{d}_t$	0.7442mm	0.8350mm	0.8293mm
		$\overline{d}_\%$	0.1924%	0.2078%	0.2121%
		$\overline{T}_{\boldsymbol{tt}}$	$3.5 \cdot 10^{-11}$	$2.1 \cdot 10^{-11}$	$2.4 \cdot 10^{-12}$
$\sigma_p = 0.1$pxl	$s = 87.80\%$	$\overline{r}$	0.1780mm	0.0942mm	0.0820mm
		$\overline{d}_\alpha$	0.1548°	0.2626°	0.2594°
		$\overline{d}_t$	3.3287mm	2.9453mm	2.9000mm
		$\overline{d}_\%$	0.8220%	0.7735%	0.7623%
		$\overline{T}_{\boldsymbol{tt}}$	1288.9	21.314	2.1993
$\sigma_p = 0.5$pxl	$s = 92.68\%$	$\overline{r}$	0.5628mm	0.4805mm	0.3360mm
		$\overline{d}_\alpha$	0.6901°	1.3136°	1.1805°
		$\overline{d}_t$	14.502mm	15.232mm	15.009mm
		$\overline{d}_\%$	3.3303%	3.7072%	3.4587%
		$\overline{T}_{\boldsymbol{tt}}$	3583.1	735.16	60.674

Table 5.1.4.: Simulated fisheye camera. Error values for the plane poses when executing the initial calibration procedure for 41 different configurations of 5 planes.

			before BA	after BA	after refinement
$\sigma_p = 10^{-7}$pxl	$s = 55.00\%$	$\overline{r}$	9638.9mm	0.0703mm	0.0670mm
		$\overline{d}_\alpha$	11.412°	0.2431°	0.2439°
		$\overline{d}_t$	322.48mm	3.36568mm	2.9941mm
		$\overline{d}_\%$	53.161%	0.8189%	0.8481%
		$\overline{T}_{\boldsymbol{tt}}$	$8.3 \cdot 10^{-8}$	$1.4 \cdot 10^{-10}$	$2.9 \cdot 10^{-12}$
$\sigma_p = 0.1$pxl	$s = 60.00\%$	$\overline{r}$	958.39mm	0.1981mm	0.1708mm
		$\overline{d}_\alpha$	11.332°	0.3653°	0.4666°
		$\overline{d}_t$	308.63mm	6.9946mm	8.2778mm
		$\overline{d}_\%$	53.461%	1.2111%	1.50518%
		$\overline{T}_{\boldsymbol{tt}}$	1655.8	116.97	10.178
$\sigma_p = 0.5$pxl	$s = 50.00\%$	$\overline{r}$	1078.2mm	0.8955mm	0.7488mm
		$\overline{d}_\alpha$	11.259°	2.6503°	2.5861°
		$\overline{d}_t$	329.23mm	35.201mm	33.416mm
		$\overline{d}_\%$	58.287%	6.9709%	6.8818%
		$\overline{T}_{\boldsymbol{tt}}$	$2.8 \cdot 10^4$	$2.9 \cdot 10^3$	54.593

Table 5.1.5.: Simulated non-central conic catadioptric camera. Error values for the plane poses when executing the initial calibration procedure for 20 different configurations of 5 planes.

	fisheye	catadioptric
proposed	0.2083mm	0.3883mm
Scaramuzza	0.4030mm	0.9840mm

Table 5.2.1.: Residual average ray-point distances for the two imaging systems calibrated in [RW12a].

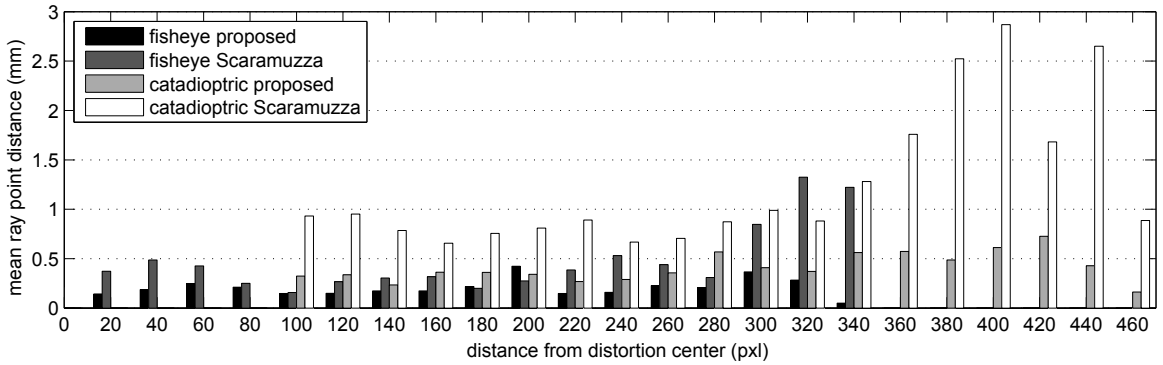


Figure 5.2.1.: Radial distribution of the residual ray-point distances with respect to the distortion center (from [RW12a]).

method was tested with two different imaging systems: a fisheye camera and a non-central catadioptric system composed of an unknown omnidirectional mirror and a perspective camera. For comparison, both systems were also calibrated with the approach proposed by Scaramuzza et al. in [SMS06], which was proven by Puig et al. in [PBSG11] to deliver good results for similar setups.
The comparison shows that the proposed calibration procedure performs better for both cameras. The residual ray-point distance is reduced significantly with respect to the values from Scaramuzza's method (confer Table 5.2.1). When regarding the radial distribution of the errors it is noteworthy that the values for Scaramuzza's approach increase with the distance from the distortion center determined by his method. This indicates that the used polynomial function is not appropriate for modeling the severe distortions at the image borders. For the proposed method, this effect is less obvious (see Figure 5.2.1). Figure 5.2.2 shows the values of the final ray-point distances for the complete image after calibrating with the proposed generic approach.
Although these results are promising, there are some "inconveniences" that occur during the calibration procedure. In fact, a lot of manual interaction and many iterations are required to achieve the shown results. Sometimes the pose estimation for new views fails or the bundle adjustment diverges for no obvious reason. This can be remedied by selecting different viewing rays for the optimization processes or by manually setting the order in which new views are added, a method which can hardly be called user-friendly or even scientifically sound. Furthermore, the results reveal that the calibration accuracy varies within the image. As can be seen in Figure 5.2.2 (right), for example, the residual errors

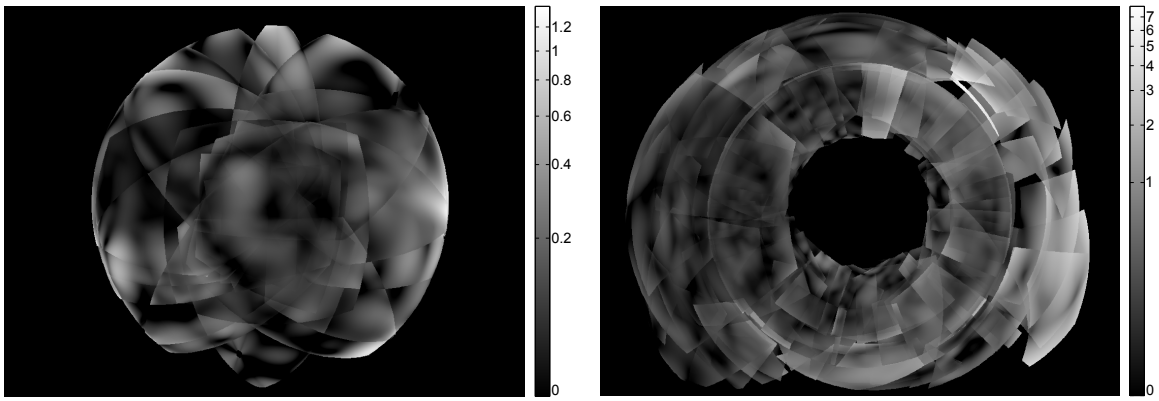


Figure 5.2.2.: Residual ray-point distances in mm after calibrating the fisheye camera (left) and the catadioptric camera (right) with the procedure proposed in [RW12a].

on the right side of the catadioptric camera tend to be bigger than those on the left side. This important information is ignored when the surface model is created and therefore lost to the user of the final camera model. Subsequent tasks could use information about the accuracy (or more precisely said: the inaccuracy or uncertainty) of viewing rays to assess the quality of their results.
To overcome these problems, the procedure described before is modified to make it more robust and also incorporate uncertainty information. This includes the pose determination of additional calibration planes to expand the calibrated image region. The improved version is described and evaluated in the following section. In Section 5.2.2, the results for a complete calibration of two simulated camera models are shown. Real cameras will be calibrated in Chapter 6.

5.2.1. Plane pose from uncertain viewing rays

The essential procedure for expanding the calibrated image region is the determination of poses of new calibration planes, which cover additional areas in the camera image (see Figure 3.3.3 for an illustration). As also non-central cameras are to be included, standard procedures for pose estimation based on projective geometry cannot be used. Instead, a linear approach for the estimation of plane poses from arbitrary viewing rays was proposed in [RW12a]. It will be augmented here to increase its robustness. Furthermore, uncertainty information will be used during the procedure and therefore provide covariance matrices for the results.
The pose of a plane can be determined if the parameters of selected viewing rays that intersect it and the corresponding local intersections are known. As stated before, when a point $\mathbf{P}$ lies on a line $\mathbf{L} = (\boldsymbol{d}^T, \boldsymbol{m}^T)^T = (L_1, L_2, L_3, L_4, L_5, L_6)^T$, the following equation

is fulfilled:

$$\overline{\Gamma}(\mathbf{L})\mathbf{P} = \mathbf{0}.$$

$\overline{\Gamma}(\mathbf{L})$ is called the dual Plücker matrix of $\mathbf{L}$ and can be constructed via

$$\overline{\Gamma}(\mathbf{L}) = \begin{bmatrix} \mathrm{S}(\boldsymbol{d}) & -\boldsymbol{m} \\ \boldsymbol{m}^T & 0 \end{bmatrix}. \tag{5.2.1}$$

The local intersection points $\boldsymbol{p}_{n,i} = (p_{n,i,1}, p_{n,i,2})^T$ of viewing ray $\mathbf{L}_i$ with the new plane n can be determined by using the interpolation abilities of the spline surface that describes the chessboard pattern in the corresponding image. The rays are given in the camera coordinate system. Transferring the $\boldsymbol{p}_{n,i}$ to this system is done with the help of the desired unknown transformation

$${}^1\mathrm{T}_n = \begin{bmatrix} \mathrm{R}_n & \boldsymbol{t}_n \\ \mathbf{0}_3^T & 1 \end{bmatrix}. \tag{5.2.2}$$

This leads to the model equation

$$\overline{\Gamma}(\mathbf{L}_i) \cdot {}^1\mathrm{T}_n \cdot \mathbf{P}_{n,i} = \mathbf{0}_4 \tag{5.2.3}$$

where $\mathbf{P}_{n,i} = (p_{n,i,1}, p_{n,i,2}, 0, 1)^T = (\boldsymbol{p}_{n,i}^{3D\,T}, 1)^T$ is the homogeneous 3D representation of the intersection point. From now on, the indices n and i will be dropped for better readability. Combining (5.2.3) with (5.2.1) and (5.2.2) gives

$$\begin{aligned} & \begin{bmatrix} \mathrm{S}(\boldsymbol{d}) & -\boldsymbol{m} \\ \boldsymbol{m}^T & 0 \end{bmatrix} \begin{bmatrix} \mathrm{R} & \boldsymbol{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{pmatrix} \boldsymbol{p}^{3D} \\ 1 \end{pmatrix} = \mathbf{0}_4 \\ \Leftrightarrow \quad & \begin{bmatrix} \mathrm{S}(\boldsymbol{d}) & -\boldsymbol{m} \\ \boldsymbol{m}^T & 0 \end{bmatrix} \begin{pmatrix} \mathrm{R}\boldsymbol{p}^{3D} + \boldsymbol{t} \\ 1 \end{pmatrix} = \mathbf{0}_4. \end{aligned}$$

The first three rows are used to define the model function

$$\boldsymbol{f}(\boldsymbol{x}) = \mathrm{S}(\boldsymbol{d}) \left(\mathrm{R}\boldsymbol{p}^{3D} + \boldsymbol{t}\right) \stackrel{!}{=} \boldsymbol{m}. \tag{5.2.4}$$

Because the third element of $\boldsymbol{p}^{3D}$ is zero, the third column of the rotation matrix is not part of the parameter vector, which is consequently defined by the remaining elements of the desired transformation:

$$\boldsymbol{x} = (r_{11}, r_{12}, r_{21}, r_{22}, r_{31}, r_{32}, t_1, t_2, t_3)^T.$$

Rewriting $\boldsymbol{f}(\boldsymbol{x})$ gives

$$\begin{bmatrix} 0 & L_3 & -L_2 \\ -L_3 & 0 & L_1 \\ L_2 & -L_1 & 0 \end{bmatrix} \begin{pmatrix} r_{11}p_1 + r_{12}p_2 + t_1 \\ r_{21}p_1 + r_{22}p_2 + t_2 \\ r_{31}p_1 + r_{32}p_2 + t_3 \end{pmatrix} = \begin{pmatrix} L_4 \\ L_5 \\ L_6 \end{pmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 0 & 0 & L_3p_1 & L_3p_2 & -L_2p_1 & -L_2p_2 & 0 & L_3 & -L_2 \\ -L_3p_1 & -L_3p_2 & 0 & 0 & L_1p_1 & L_1p_2 & -L_3 & 0 & L_1 \\ L_2p_1 & L_2p_2 & -L_1p_1 & -L_1p_2 & 0 & 0 & L_2 & -L_1 & 0 \end{bmatrix} \boldsymbol{x} = \begin{pmatrix} L_4 \\ L_5 \\ L_6 \end{pmatrix}.$$

The dual Plücker matrix $\overline{\Gamma}$ only has rank 2. Therefore, two rows need to be picked to get linear independent equations for the final equation system. This is done here by selecting those rows that contain the biggest absolute values, leading to a row-reduced equation system

$$\mathrm{A}_i^* \boldsymbol{x} = \boldsymbol{b}_i^*$$

for each viewing ray i. As there are 9 unknowns, at least $n_L = 5$ different rays are needed to solve for the parameter vector. Stacking the corresponding matrices and vectors gives the final equation system

$$\underbrace{\begin{bmatrix} \mathrm{A}_1^* \\ \vdots \\ \mathrm{A}_{n_L}^* \end{bmatrix}}_{\mathrm{A}} \boldsymbol{x} = \underbrace{\begin{bmatrix} \boldsymbol{b}_1^* \\ \vdots \\ \boldsymbol{b}_{n_L}^* \end{bmatrix}}_{\boldsymbol{b}}.$$

The vector $\boldsymbol{b}$ can be considered as a measurement vector. This allows to use the corresponding elements of the line covariance matrices $\Sigma_{\mathbf{L}_i\mathbf{L}_i}$ to construct the adjusted covariance matrices $\Sigma_{\boldsymbol{b}_i^*\boldsymbol{b}_i^*}$. They, in turn, serve to create the measurement covariance matrix

$$\Sigma_{\boldsymbol{b}\boldsymbol{b}} = \begin{bmatrix} \Sigma_{\boldsymbol{b}_1^*\boldsymbol{b}_1^*} & & \\ & \ddots & \\ & & \Sigma_{\boldsymbol{b}_{n_L}^*\boldsymbol{b}_{n_L}^*} \end{bmatrix}$$

which is inverted to obtain a weight matrix. This delivers the weighted least squares solution for the parameter vector

$$\boldsymbol{x} = (\mathrm{A}^T \Sigma_{\boldsymbol{b}\boldsymbol{b}}^{-1} \mathrm{A})^{-1} \mathrm{A}^T \Sigma_{\boldsymbol{b}\boldsymbol{b}}^{-1} \boldsymbol{b}$$

which has the uncertainties

$$\Sigma_{\boldsymbol{x}\boldsymbol{x}} = (\mathrm{A}^T \Sigma_{\boldsymbol{b}\boldsymbol{b}}^{-1} \mathrm{A})^{-1}.$$

The elements of $\boldsymbol{x}$ serve to construct the two vectors $\boldsymbol{r}'_1 = (x_1, x_3, x_5)^T$ and $\boldsymbol{r}'_2 = (x_2, x_4, x_6)^T$, their covariance matrices $\Sigma_{\boldsymbol{r}'_1\boldsymbol{r}'_1}$ and $\Sigma_{\boldsymbol{r}'_2\boldsymbol{r}'_2}$ are extracted from $\Sigma_{\boldsymbol{x}\boldsymbol{x}}$. As no further constraints are used that enforce unit length or orthogonality, these results need to be transformed to get valid rotation vectors $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and also $\boldsymbol{r}_3$ for constructing R_n. First, normalization gives

$$\boldsymbol{r}_1 = \frac{\boldsymbol{r}'_1}{|\boldsymbol{r}'_1|}, \quad \Sigma_{\boldsymbol{r}_1\boldsymbol{r}_1} = \mathrm{J}_{r_1} \Sigma_{\boldsymbol{r}'_1\boldsymbol{r}'_1} \mathrm{J}_{r_1}^T$$

with

$$\mathrm{J}_{r_1} = (\mathrm{I}_3 - \boldsymbol{r}_1^T \boldsymbol{r}_1)/|\boldsymbol{r}'_1|. \tag{5.2.5}$$

$\boldsymbol{r}_1$ and $\boldsymbol{r}'_2$ span a plane, which has $\boldsymbol{r}_3$ as its normal vector:

$$\boldsymbol{r}'_3 = \boldsymbol{r}_1 \times \boldsymbol{r}'_2.$$

$\Sigma_{\boldsymbol{r}_1\boldsymbol{r}_1}$ and $\Sigma_{\boldsymbol{r}'_2\boldsymbol{r}'_2}$ allow to calculate $\Sigma_{\boldsymbol{r}'_3\boldsymbol{r}'_3}$. Subsequent normalization gives

$$\boldsymbol{r}_3 = \frac{\boldsymbol{r}'_3}{|\boldsymbol{r}'_3|} \quad \text{and} \quad \Sigma_{\boldsymbol{r}_3\boldsymbol{r}_3} = \mathrm{J}_{r_3}\Sigma_{\boldsymbol{r}'_3\boldsymbol{r}'_3}\mathrm{J}_{r_3}^T \quad (\mathrm{J}_{r_3} \text{ as shown in (5.2.5)}).$$

Now two of the three rotational vectors are determined. The last one is calculated by their cross product:

$$\boldsymbol{r}_2 = \boldsymbol{r}_3 \times \boldsymbol{r}_1$$

with $\Sigma_{\boldsymbol{r}_2\boldsymbol{r}_2}$ resulting from $\Sigma_{\boldsymbol{r}_3\boldsymbol{r}_3}$ and $\Sigma_{\boldsymbol{r}_1\boldsymbol{r}_1}$. Now, the desired rotation matrix can be constructed as $\mathrm{R}_n = [\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{r}_3]^T$.
Although the elements of the plane position $\boldsymbol{t}_n$ are also part of the parameter vector, experiments showed that this method is not robust enough to get acceptable results for t_1, t_2 and t_2. In fact, the actual choice of rays has a big influence on the overall accuracy, which effects the translational elements much more than the rotational ones. Therefore, a second procedure is executed that recalculates the plane position while assuming the rotation matrix to be known. It uses the same approach as before, but utilizes the model function (5.2.4) in a different way, as now only the vector $\boldsymbol{t}$ is considered to be unknown:

$$\begin{aligned} & \mathrm{S}(\boldsymbol{d})\left(\mathrm{R}\boldsymbol{p}^{3D} + \boldsymbol{t}\right) = \boldsymbol{m} \\ \Leftrightarrow \quad & \underbrace{\mathrm{S}(\boldsymbol{d})}_{\mathrm{A}_t}\boldsymbol{t} = \underbrace{\boldsymbol{m} - \mathrm{S}(\boldsymbol{d})\mathrm{R}\boldsymbol{p}^{3D}}_{\boldsymbol{b}_t}. \end{aligned}$$

Again, the right side is interpreted as a measurement vector. The corresponding covariance matrix is determined as follows:

$$\Sigma_{\boldsymbol{b}_t\boldsymbol{b}_t} = \mathrm{J}_m\Sigma_{\boldsymbol{m}\boldsymbol{m}}\mathrm{J}_m^T + \mathrm{J}_d\Sigma_{\boldsymbol{d}\boldsymbol{d}}\mathrm{J}_d^T + \mathrm{J}_R\Sigma_{\boldsymbol{R}\boldsymbol{R}}\mathrm{J}_R^T + \mathrm{J}_p\Sigma_{\boldsymbol{p}\boldsymbol{p}}\mathrm{J}_p^T.$$

The needed Jacobians are

$$\begin{aligned} \mathrm{J}_m &= \frac{\partial \boldsymbol{b}_t}{\partial \boldsymbol{m}} = \mathrm{I}_3 \\ \mathrm{J}_d &= \frac{\partial \boldsymbol{b}_t}{\partial \boldsymbol{d}} = -\mathrm{S}(\mathrm{R}\boldsymbol{p}^{3D}) \\ \mathrm{J}_R &= \frac{\partial \boldsymbol{b}_t}{\partial \boldsymbol{R}} = -\mathrm{S}(\boldsymbol{d})\frac{\partial}{\partial \boldsymbol{R}}\mathrm{R}\boldsymbol{p}^{3D} \overset{(2.1.27)}{=} \mathrm{S}(\boldsymbol{d}) \cdot \mathrm{R} \cdot \mathrm{S}(\boldsymbol{p}^{3D}) \\ \mathrm{J}_p &= \frac{\partial \boldsymbol{b}_t}{\partial (p_1, p_2)^T} = -\mathrm{S}(\boldsymbol{d}) \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{bmatrix}. \end{aligned}$$

In the current model equation, $\mathrm{A}_t = \mathrm{S}(\boldsymbol{d})$ is rank deficient. Therefore, two linearly independent rows are selected as before, forming the row-reduced system

$$\mathrm{A}^*_{t,i}\boldsymbol{t} = \boldsymbol{b}^*_{t,i}.$$

The covariance matrix $\Sigma_{\boldsymbol{b}_t\boldsymbol{b}_t}$ of the right side has to be adjusted accordingly to get $\Sigma_{\boldsymbol{b}^*_{t,i}\boldsymbol{b}^*_{t,i}}$. In this case, only two rays are needed to solve for $\boldsymbol{t}$. Nevertheless, the same amount as before will be used, giving a system with $2n_L$ equations:

$$\underbrace{\begin{bmatrix} \mathrm{A}^*_{t,1} \\ \vdots \\ \mathrm{A}^*_{t,n_L} \end{bmatrix}}_{\mathrm{A}} \boldsymbol{t} = \underbrace{\begin{bmatrix} \boldsymbol{b}^*_{t,1} \\ \vdots \\ \boldsymbol{b}^*_{t,n_L} \end{bmatrix}}_{\boldsymbol{b}} \quad \text{with} \quad \Sigma_{\boldsymbol{b}\boldsymbol{b}} = \begin{bmatrix} \Sigma_{\boldsymbol{b}^*_{t,1}\boldsymbol{b}^*_{t,1}} & & \\ & \ddots & \\ & & \Sigma_{\boldsymbol{b}^*_{t,n_L}\boldsymbol{b}^*_{t,n_L}} \end{bmatrix}.$$

As before, the weighted least squares approach serves to get the desired solution and its covariance matrix:

$$\boldsymbol{t} = (\mathrm{A}^T\Sigma^{-1}_{\boldsymbol{b}\boldsymbol{b}}\mathrm{A})^{-1}\mathrm{A}^T\Sigma^{-1}_{\boldsymbol{b}\boldsymbol{b}}\boldsymbol{b}, \quad \Sigma_{\boldsymbol{t}\boldsymbol{t}} = (\mathrm{A}^T\Sigma^{-1}_{\boldsymbol{b}\boldsymbol{b}}\mathrm{A})^{-1}.$$

At this point, a final check on the conclusiveness of the solution is advisable. In case of a central camera, there are in fact two valid solutions, lying on opposite sides of the optical center. To verify that the correct one was calculated, the direction vector from the optical center (or, more generally speaking, the caustic center) to the determined position $\boldsymbol{t}$ can be compared to the average direction of all viewing rays used in the procedure. If they point in opposite directions, the false solution was obtained. This can be remedied by rotating around the caustic center, or, which is more exact, by inverting the rotation vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ and repeating the calculation of the translation vector.
To put the system into an adequate state for testing the procedure of plane pose estimation, the initial calibration procedure from the last section is executed again. Afterwards, the poses of the involved planes are set to their ground truth values, but the covariance matrices from the initial calibration are kept. This allows to evaluate the performance of the pose estimation procedure under realistic uncertainty conditions, while the input data is perfectly correct. Now, a new plane is selected from the set of all calibration views by identifying the one which has the most overlap with the currently calibrated area. Then, 18 image pixels that lie within this overlap are selected randomly. This leads to a redundancy of 4 for the least squares problem. With the uncertain rays from these pixel positions and the corresponding local intersection points with the additional plane, the pose estimation procedure is started. After that, the bundle adjustment and the refinement process from the last section are executed. After each step, the error values which contribute to $\overline{r}$, $\overline{d}_t$, $\overline{d}_{\%}$, $\overline{d}_\alpha$ and $\overline{T}_{\boldsymbol{t}\boldsymbol{t}}$ are determined as before. This evaluation procedure is executed as before for a central fisheye camera and a non-central catadioptric camera with a conic mirror. Different noise levels of the simulated chessboard corners show the influence of measurement noise on the whole calibration procedure.

			before BA	after BA	after refinement
$\sigma_p = 10^{-7}$pxl	$s = 87.80\%$	$\overline{r}$	0.5350mm	0.0214mm	0.0165mm
		$\overline{d}_\alpha$	4.8401°	0.1819°	0.1634°
		$\overline{d}_t$	32.204mm	1.7600mm	1.7074mm
		$\overline{d}_\%$	6.7237%	0.3118%	0.3085%
		$\overline{T}_{\boldsymbol{tt}}$	257.90	$3.9 \cdot 10^{-11}$	$2.4 \cdot 10^{-12}$
$\sigma_p = 0.1$pxl	$s = 95.12\%$	$\overline{r}$	0.0989mm	0.0775mm	0.0610mm
		$\overline{d}_\alpha$	1.0492°	0.2834°	0.2828°
		$\overline{d}_t$	6.0308mm	2.3068mm	2.6492mm
		$\overline{d}_\%$	0.9048%	0.4643%	0.5204%
		$\overline{T}_{\boldsymbol{tt}}$	62.043	62.042	1.8291
$\sigma_p = 0.5$pxl	$s = 97.56\%$	$\overline{r}$	0.9884mm	0.3698mm	0.2510mm
		$\overline{d}_\alpha$	3.8183°	1.5447°	1.2158°
		$\overline{d}_t$	24.182mm	10.178mm	8.6241mm
		$\overline{d}_\%$	3.4780%	1.9315%	1.5704%
		$\overline{T}_{\boldsymbol{tt}}$	909.04	199.64	37.026

Table 5.2.2.: Simulated fisheye camera. Accuracy of additional planes for 41 different configurations.

The results of the experiments are shown in Tables 5.2.2 and 5.2.3 for the simulated fisheye and the simulated conic catadioptric camera, respectively. They reveal that the bundle adjustment procedure is able to compensate for high errors in the initial pose estimates, leading to final values that are very similar to those of the initial calibration procedure (compare Tables 5.1.4 and 5.1.5). Almost all experiments deliver a result, which is indicated by high success rates around 90% for the fisheye and from 70% to 95% for the catadioptric camera. This proves that the proposed procedure for plane pose estimation from viewing rays is able to determine good initial values for both cameras, allowing to use the method to expand the calibrated image region.

5.2.2. Complete calibration results

The last section showed how the calibrated image region can be expanded by determining the poses of additional calibration planes based on already known viewing rays. Executing this method for all planes finally leads to a completely calibrated camera, i.e. a camera where most pixels "see" at least two planes, facilitating the determination of the corresponding viewing ray. The procedure of complete calibration is the following:

1. Take images of a calibration plane in different poses, such as each pixel is at least

			before BA	after BA	after refinement
$\sigma_p = 10^{-7}$pxl	$s = 70.00\%$	$\overline{r}$	0.3231mm	0.04971mm	0.0448mm
		$\overline{d}_\alpha$	3.6013°	0.1746°	0.2117°
		$\overline{d}_t$	12.896mm	2.6332mm	2.9041mm
		$\overline{d}_\%$	1.9103%	0.4568%	0.5086%
		$\overline{T}_{\boldsymbol{tt}}$	0.8143	$7.6 \cdot 10^{-11}$	$5.6 \cdot 10^{-12}$
$\sigma_p = 0.1$pxl	$s = 95.00\%$	$\overline{r}$	0.4238mm	0.2491mm	0.1794mm
		$\overline{d}_\alpha$	0.8439°	0.6516°	0.5367°
		$\overline{d}_t$	7.0874mm	6.0867mm	5.8252mm
		$\overline{d}_\%$	1.0136%	0.8682%	0.8363%
		$\overline{T}_{\boldsymbol{tt}}$	14.663	6.7198	2.8920
$\sigma_p = 0.5$pxl	$s = 85.00\%$	$\overline{r}$	1.1414mm	0.9186mm	0.7133mm
		$\overline{d}_\alpha$	3.7176°	1.4837°	1.4294°
		$\overline{d}_t$	33.813mm	23.259mm	20.138mm
		$\overline{d}_\%$	3.5850%	2.5040%	2.1639%
		$\overline{T}_{\boldsymbol{tt}}$	42.697	211.843	57.023

Table 5.2.3.: Simulated non-central conic catadioptric camera. Accuracy of additional planes for 20 different configurations.

covered twice.

2. Initial calibration: pick one calibration plane as virtual image plane and execute the linear calibration procedure described in Section 5.1.3. Then, minimize the ray-point distance (Section 5.1.6) and refine the results (Section 5.1.6.3).

3. Select that additional calibration image which has the biggest overlap with the currently calibrated image region.

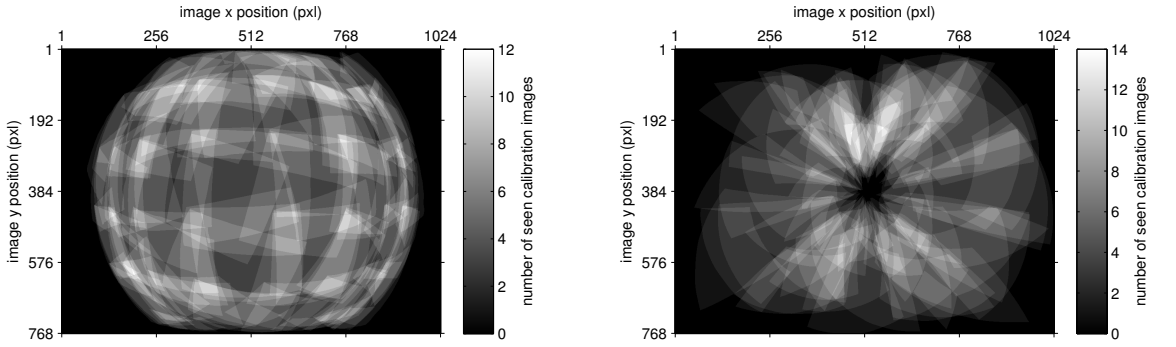
4. Determine the pose of the additional plane by using the method described in Section 5.2.1.

5. Choose a set of $V - 1$ calibrated images that overlap with the additional one and minimize the ray-point distance by adjusting the plane poses. In case the virtual image plane is not one of them: select the plane with the lowest uncertainty as static plane.

6. If there are uncalibrated planes left: go back to 2.

7. Execute bundle adjustment for all planes as a final refinement.

This method is executed for the same simulated cameras as before: a central fisheye camera and a non-central conic catadioptric system. Calibration plane poses are generated lying randomly on three spheres with different radii around the caustic centers. The main goal is to get at least three planes seen at each pixel position, but the actual number is much higher in some areas (up to 12 for the fisheye and 14 for the catadioptric camera). The final coverage of the camera images from all calibration planes can be found in Figure 5.2.3. All pixels that have a value bigger than 1 contribute to the calibrated region. The resulting final error values can be found in Table 5.2.4. In this case, they are calculated over all calibration planes except the virtual image plane. The experiments are conducted for two different chessboard corner noise levels. The very low value of $\sigma_p = 10^{-7}$pxl gives an idea about the accuracy that could be expected with an almost perfect corner detector, $\sigma_p = 0.1$pxl is a realistic value which hints at possible real-world results. For the fisheye camera, the absolute values for residual ray-point distance as well as positional and angular error of the planes are slightly increased in comparison to the results of the initial calibration procedure from Table 5.1.4. This is not surprising as many more planes are involved, including those that lie further from the camera. Furthermore, the actual field of view is much bigger, getting close to 180°. Some of the planes have no overlap with the virtual image plane. This means they are not directly connected to the fixed coordinate system, leading to accumulated errors. Nevertheless, the relative accuracy as measured by $\overline{d}_{\%}$ is even smaller than after the initial procedure. The values of 0.1019% for the low noise level and 0.2870% with σ_p=0.1pxl prove that the procedure of complete calibration can deliver highly accurate final results.
For the non-central conic catadioptric system, the average positional error $\overline{d}_t$ lies at 7.7mm and is therefore higher than for the fisheye camera. This corresponds to a relative positional accuracy of almost 0.7%. The angular error $\overline{d}_\alpha$ is close to 0.2° for the low noise level and a little higher than 0.5° for the realistic noise level.
The distributions of the residual ray-point distances in the camera images can be regarded in Figure 5.2.4. They reveal that the ray accuracy varies significantly within the calibrated area. Overall means are higher than the $\overline{r}$ in Table 5.2.4 which only include the rays used for the optimization procedure. Nevertheless, they still lie well below half a millimeter for both systems and noise levels. These means also include outliers, which can have very high ray-point distances, e.g. up to 12.8mm for the catadioptric camera with realistic noise level or even 46.1mm for the low noise level. Additionally, another aspect becomes obvious when regarding the results for the catadioptric system. As the viewing field is horizontally omnidirectional, the calibration planes do not lie on hemispheres as for the fisheye camera, but on rings around the camera. Therefore, the calibration procedure starts at one position and then expands in both directions along these rings. During the procedure, plane estimation errors accumulate. This leads to comparably high ray-point distances in that area where the ring finally closes, as can be seen on the left side for the noise level of 10^{-7}pxl and at the bottom for $\sigma_p = 0.1$pxl. Experiments show that the final bundle adjustment for all planes can diminish these effects, but not completely erase them.



(a) Simulated fisheye camera (115 planes).

(b) Simulated conic catadioptric camera (87 planes).

Figure 5.2.3.: Image coverage after the complete calibration procedure.

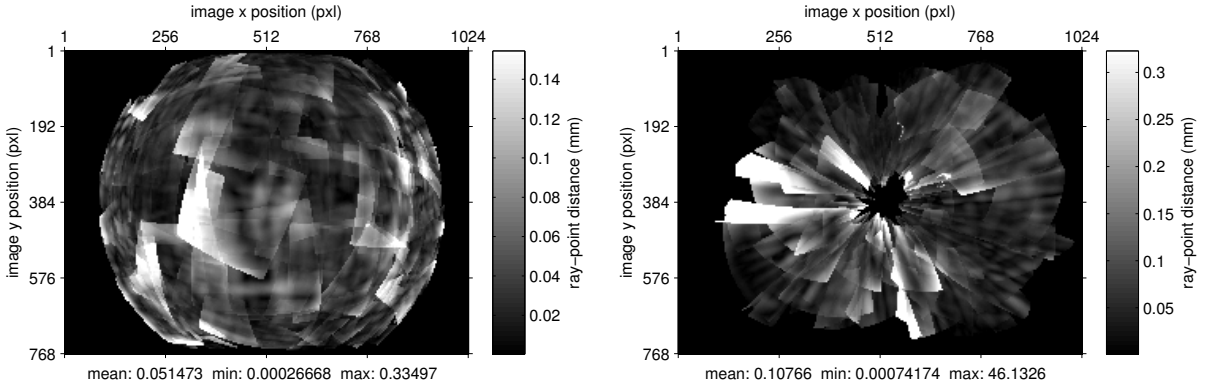
	fisheye (114 planes)		catadioptric (86 planes)	
σ_p	10^{-7}pxl	0.1pxl	10^{-7}pxl	0.1pxl
$\overline{r}$	0.0233mm	0.1744mm	0.0304mm	0.3289mm
$\overline{d}_\alpha$	0.1446°	0.6709°	0.1944°	0.5107°
$\overline{d}_t$	0.9451mm	2.7493mm	2.4729mm	7.7346mm
$\overline{d}_\%$	0.1019%	0.2870%	0.2133%	0.6986%
$\overline{T}_{\boldsymbol{tt}}$	$1.1 \cdot 10^{-11}$	11.260	$1.2 \cdot 10^{-11}$	19.608

Table 5.2.4.: Final error values after the complete calibration of two different simulated imaging systems.

This concludes the evaluation of the complete calibration process. It was shown that the proposed procedure is able to determine the poses of all involved calibration planes for central as well as non-central cameras with high accuracy. The next step is to build the final surface model. It will be described in the following section, which also regards uncertainty aspects.

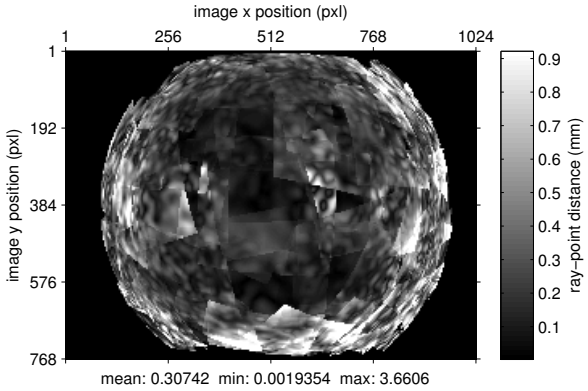
5.3. Surface model construction

In the last sections it was shown how a complete calibration of all involved calibration planes can be achieved. The final step for building the surface model, i.e. the continuous representation of the generic camera model, is to use this information to generate viewing rays which serve as data points for the corresponding 6D spline surface in Plücker coordinates (see Chapter 4). Section 4.1 proposes the general procedure for creating the spline surface model. The methods of linear extrapolation and interpolation described in

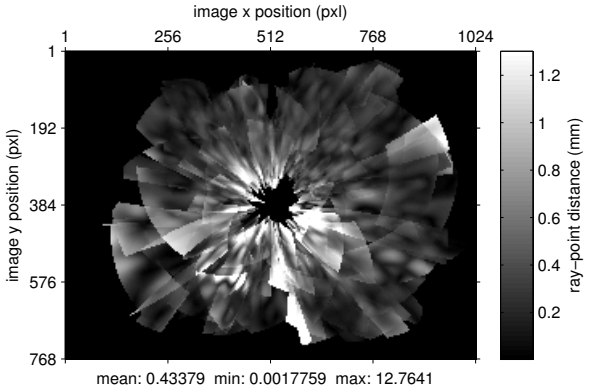


(a) Fisheye camera with $\sigma_p = 10^{-7}$pxl

(b) Catadioptric camera with $\sigma_p = 10^{-7}$pxl



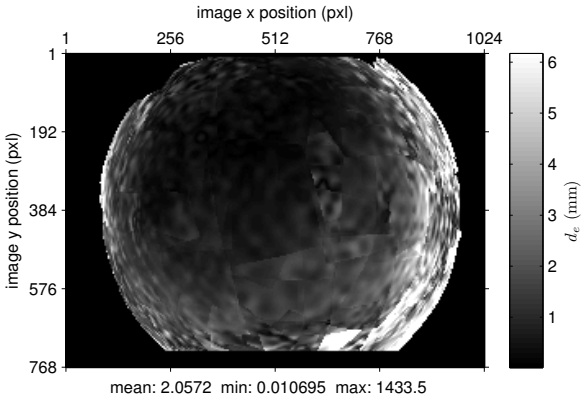
(c) Fisheye camera with $\sigma_p = 0.1$pxl



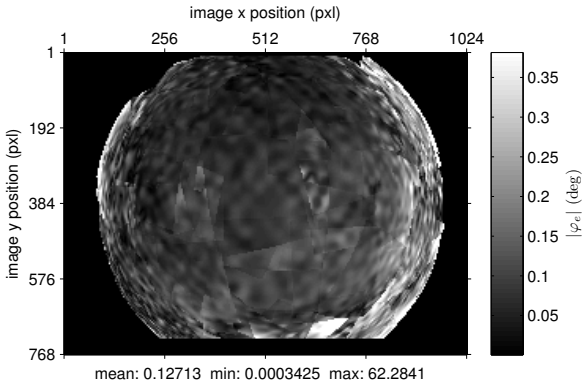
(d) Catadioptric camera with $\sigma_p = 0.1$pxl

Figure 5.2.4.: Ray-point distances after calibrating the two simulated cameras with different noise levels σ_p.

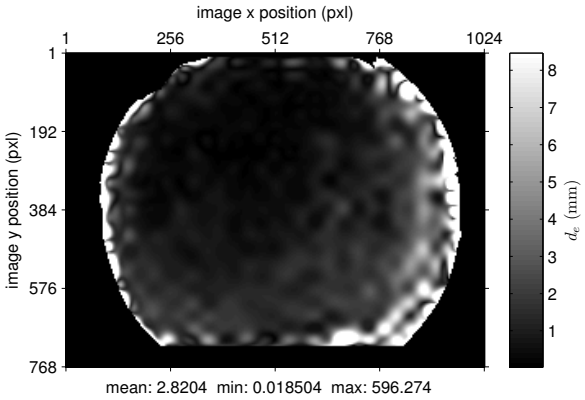
Section 4.2.1.4 are used to generate data points for pixel positions where no Plücker coordinates can be determined. Integration of the ray covariance matrices to get an uncertain surface model is done as shown in Section 4.2.3. Unlike before, where the data points were given by a simulation of the camera, the ray parameters are now determined from the local intersection points of the viewing rays with the calibration planes in a common camera coordinate system.
The global parameters of the spline surface stay the same: still the basis functions are of degree 3 and uniform parameterization is utilized. Data points are drawn on an equidistant grid with data point position distance d_d. From these, the spline control points are calculated via B-spline approximation (Section 2.3.3). The results obtained in Section 4.2.1.5 for a conic catadioptric camera with noisy data points suggest that around 30 control points are needed to get a decent model quality for an image with a resolution of 1024×768pxl (compare Figure 4.2.22). This sets an upper boundary for the data point position distance, as at least 1 data point per control point is needed. For the current camera systems, this requires d_d to be smaller than 29 pixels. It is therefore set to 28 pixels. For evaluation and comparison, the error values d_e and φ_e are determined for rays from two different sources for each pixel. First, the rays are determined directly from the intersection points with the calibration planes. This is the same procedure as for the calculation of data points for the creation of the surface model. Then, the rays are obtained from the surface model itself. Comparing the corresponding error values helps to assess the quality of the surface model. Figure 5.3.1 shows the resulting profiles of the ray distance error d_e and the absolute angular error $|\varphi_e|$ for every calibrated pixel of a simulated fisheye camera. Mean values for rays determined directly from the calibration planes lie at 2mm and 0.127°, respectively. A surface model constructed from data points with $d_d = 28$ and (30×24) control points in u and v direction has slightly increased error values of $\overline{d}_e = 2.82$mm and $|\overline{\varphi}_e| = 0.162°$. Reducing the data point position distance to 14 pixels but keeping the number of control points leads to mean error values that match those of the rays determined directly from the planes. This shows that more data points can indeed increase the model quality, as the influence of noisy data points is reduced.
Similar conclusions can be drawn from the results for the simulated non-central conic catadioptric imaging system (see Figure 5.3.2). Here, the mean positional error increases from 7.78mm to 8.34mm, the average angular error from 0.56° to 0.61° with a data point position distance of 28 pixels and (29×20) control points. As before, using more data points by setting the data point distance to 14 pixels diminishes the error values such as they are close to the ones from the calibration planes. Furthermore, as can be visually verified, ripples induced by the discontinuity in the central region (compare Section 4.2.1.4) are significantly reduced.
The values considered so far are the error values which can be determined because the ground truth data is available in these simulations. An interesting question is whether the uncertainties of the spline surface represent these errors, i.e. if the covariance matrices of the viewing rays deliver valuable information about the actual accuracy. This aspect can be analyzed by regarding the Mahalanobis distances of the ground truth rays from the



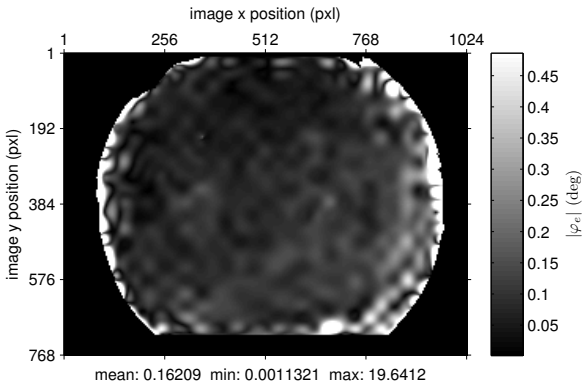
(a) Rays from calibration planes.



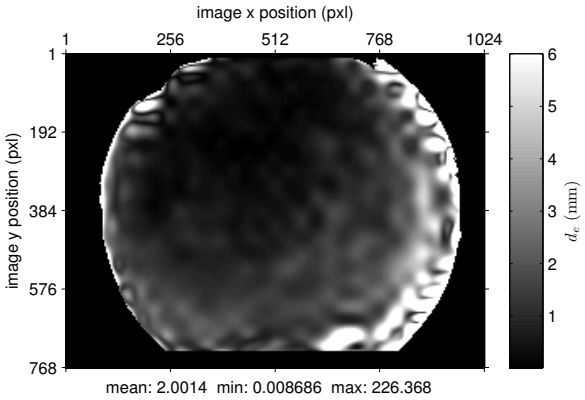
(b) Rays from calibration planes.



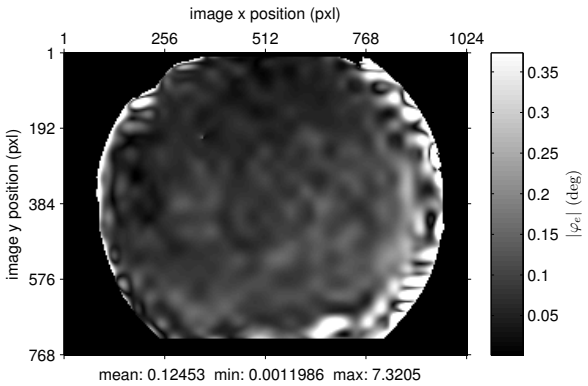
(c) Rays from a surface model ($d_d = 28$).



(d) Rays from a surface model ($d_d = 28$).

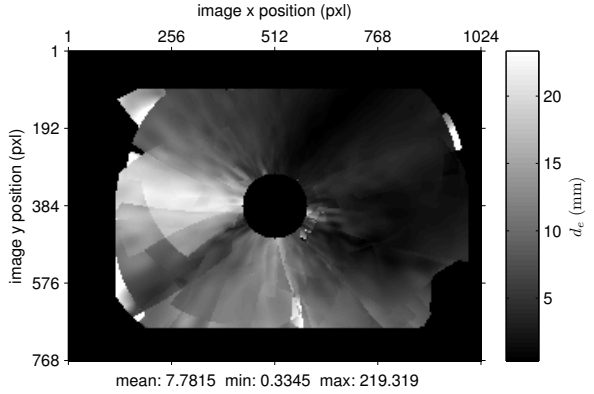


(e) Rays from a surface model ($d_d = 14$)

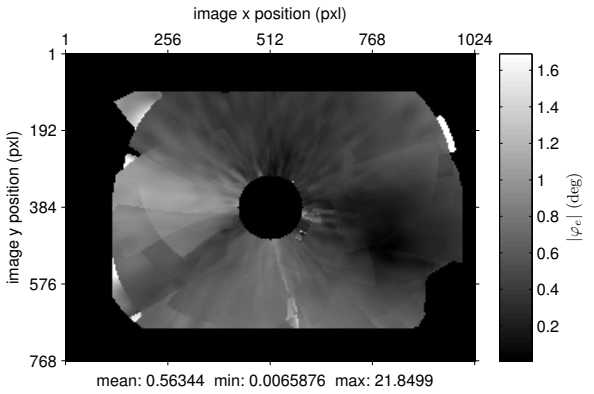


(f) Rays from a surface model ($d_d = 14$)

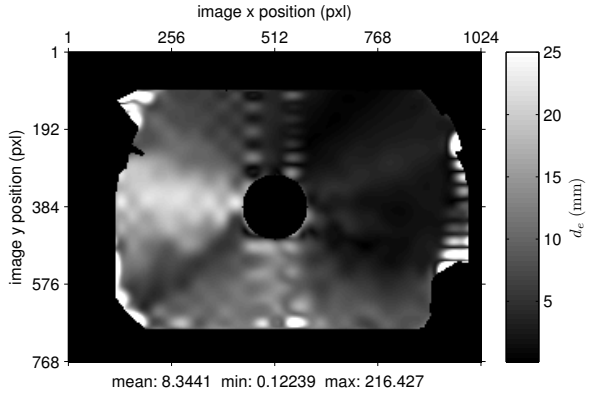
Figure 5.3.1.: Simulated fisheye camera ($\sigma_p = 0.1$pxl). Error values d_e and $|\varphi_e|$ are shown for rays determined from the calibration planes (first row) or from surface models with (30×24) control points and different data point position distances (second and third row).



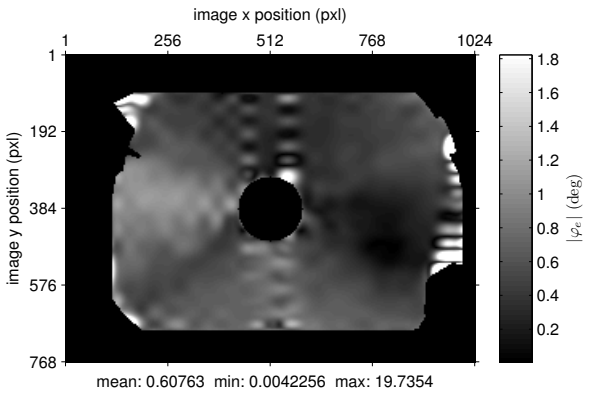
(a) Rays come from calibration planes.



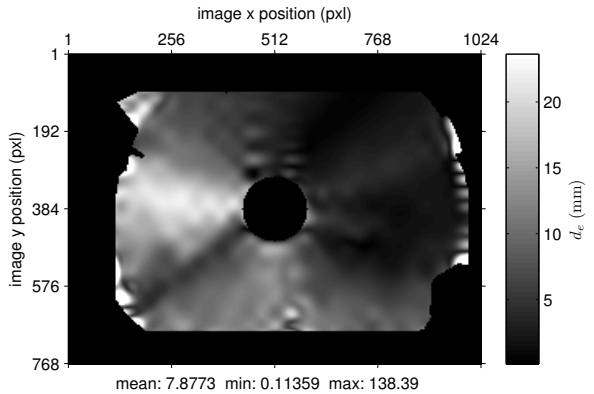
(b) Rays come from calibration planes.



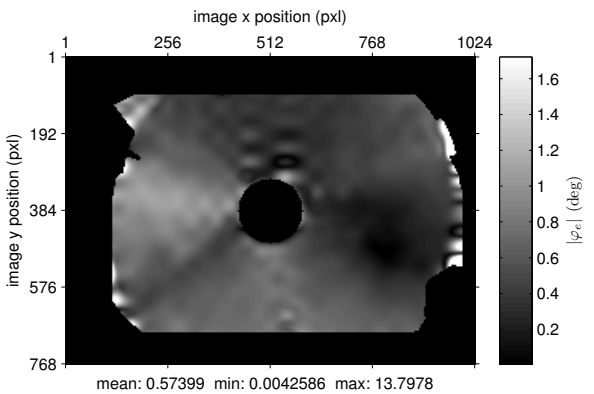
(c) Rays come from a surface model ($d_d = 28$).



(d) Rays come from a surface model ($d_d = 28$).



(e) Rays come from a surface model ($d_d = 14$).



(f) Rays come from a surface model ($d_d = 14$).

Figure 5.3.2.: Simulated catadioptric camera ($\sigma_p = 0.1$pxl). Error values d_e and $|\varphi_e|$ are shown for rays determined from the calibration planes (first row) and from surface models with (29×20) control points and different data point position distances (second and third row).

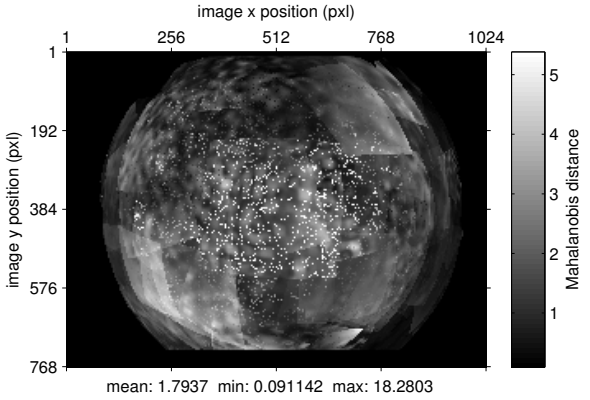
uncertain rays determined by calibration planes or surface models. Generally, the Mahalanobis distance Ω of a vector $\boldsymbol{v}_n$ from a random variable $\boldsymbol{v} \sim \mathsf{N}(\overline{\boldsymbol{v}}, \Sigma_{\boldsymbol{vv}})$ is calculated as

$$\Omega = \sqrt{(\boldsymbol{v}_n - \overline{\boldsymbol{v}})\Sigma_{\boldsymbol{vv}}^{-1}(\boldsymbol{v}_n - \overline{\boldsymbol{v}})}.$$

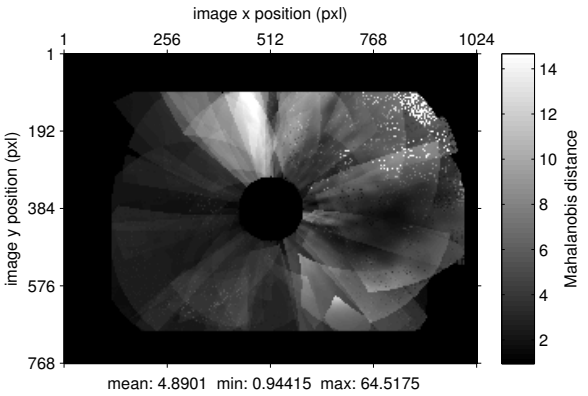
This involves inverting the covariance matrix. For a homogeneous Plücker vector, the covariance matrix is singular. Therefore, the Mahalanobis distance is calculated in reduced space (confer Section 2.4.4 for details). The resulting profiles are shown in Figure 5.3.3 for both camera systems. It is revealed that for the fisheye camera, the average Mahalanobis distance for rays from calibration planes lies at 1.79. This signifies that the ray errors are on average smaller than 2 standard deviations. Therefore, the ray uncertainties give a good indication on which errors are to be expected. For the surface model created with data point position distance d_d =28 pixels, the value is with 1.62 almost the same. When d_d is reduced to 14 pixels, however, the mean Mahalanobis distance is increased. The reason is that now more data points contribute to a single spline control point. Consequently, the expected uncertainty decreases, leading to bigger Mahalanobis distances of rays which have a similar spatial distance as before. In general, more data points are preferable, because they increase the surface model accuracy by decreasing the influence of noise and outliers. Therefore, it has to be kept in mind that when using more data points, the actual uncertainties of the rays are bigger than those estimated by the surface model. To compensate for this, it is advisable to store the mean ratio of the entries of the covariance matrices obtained from calibration planes and from the final spline surface. Multiplying covariances determined with the surface model with this factor delivers uncertainty estimates which are much more realistic.

The results for the conic catadioptric imaging system indicate that the predicted uncertainty is lower than the actual one, as the mean Mahalanobis distance of the ground truth rays lies at 4.89. This may be a consequence of the accumulated errors described in the last section. Since the uncertainties of the viewing rays are determined from the calibration planes, the corresponding covariances are small in case the relative positions of these planes are locally accurate. However, they can still be far from the absolute ground truth position, e.g. due to an accumulated offset. Therefore, it has to be kept in mind that the estimated accuracies for cameras with a mainly horizontal field of view tend to be higher than the actual ones. Apart from this aspect, using more data points increases the predicted model accuracy as before. The Mahalanobis distances with d_d = 14pxls have an average value of 9.04 and are much higher than for d_d = 28pxls (4.67). Restoring the mean ratio as a correction factor as described before leads to better uncertainty estimates of the final surface model.

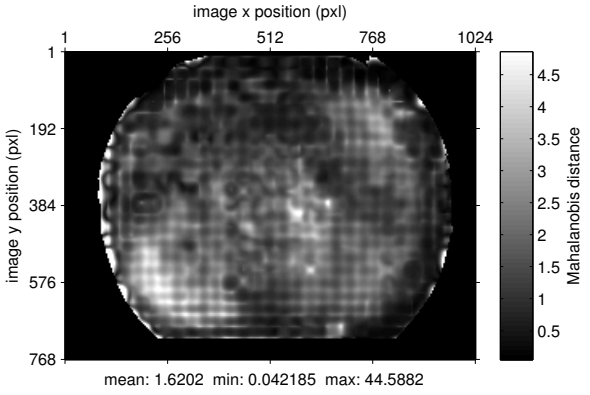
This concludes the chapter on the calibration procedure of the surface model from handheld calibration planes. It was shown that the same procedure works for central as well as non-central imaging systems. The results also prove that accurate results can be obtained, although only sparse calibration patterns are used. The estimated accuracy of the used image features (chessboard corners) can be used to deduce uncertainty information



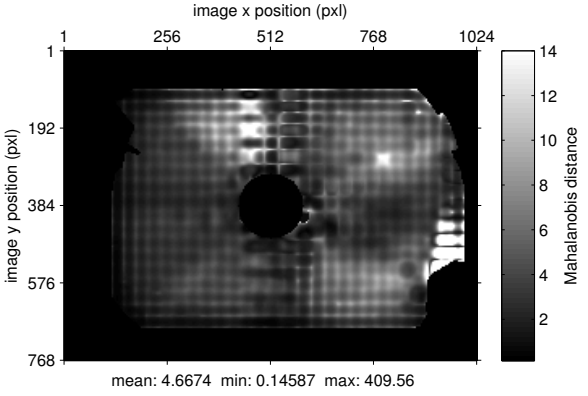
(a) Rays come from calibration planes.



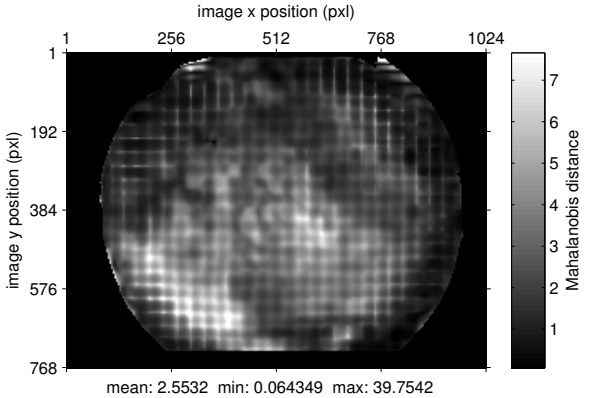
(b) Rays come from calibration planes.



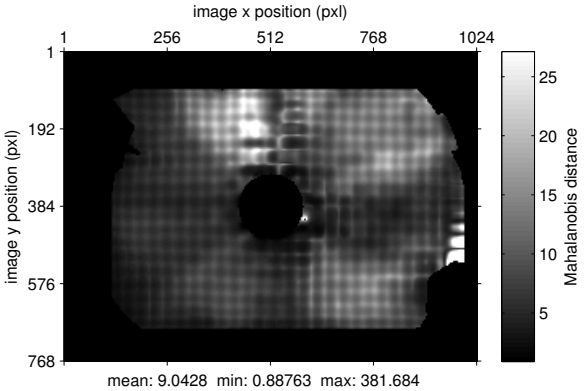
(c) Rays come from a surface model ($d_d = 28$).



(d) Rays come from a surface model ($d_d = 28$).



(e) Rays come from a surface model ($d_d = 14$).



(f) Rays come from a surface model ($d_d = 14$).

Figure 5.3.3.: Simulated fisheye (left column, (30×24) control points) and catadioptric camera (right column, 29×20 control points), $\sigma_p = 0.1$pxl. The Mahalanobis distances of simulated ground truth rays from rays are shown. These are either determined from the calibration planes (first row) or from surface models with different data point position distances (second and third row).

which proves valuable during the bundle adjustment procedure. Furthermore, rigorous uncertainty propagation allows to draw conclusions on the actual accuracy of the final surface model. The next chapter will apply the proposed method to calibrate various real imaging systems.

Chapter 6

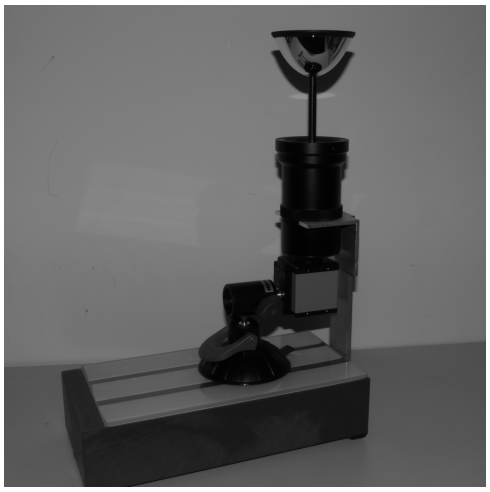
Calibration of real camera systems

The last chapter showed how the surface model as a representation of the generic camera model can be calibrated. As input data, only sparse feature points of planar calibration boards are needed. Overall, the complete image area is to be covered at least twice for each pixel. Simulations proved that the proposed procedure of complete camera calibration, which is composed of an initial calibration step and a subsequent expansion of the calibrated area, serves to determine the poses of these planes in a common coordinate system. These planes, in turn, allow to calculate viewing rays for arbitrary pixels that are used as data points to parameterize a 6D spline surface. This spline is the surface model which serves as a continuous representation of the generic camera model.

6.1. Fisheye and catadioptric camera

In this section, the surface model is used to describe two different real imaging systems (see Figures 6.1.1 and 6.1.2). At first, a camera with a fisheye lens, having a field of view of approximately 180° and a correspondingly short focal distance is analyzed. The camera itself is the monochrome DMK-31BF03 from The Imaging Source with a resolution of 1024×768 pixels, the fisheye lens is the Fujinon YV2.2x1.4A-2 with the focal length set to a value of approximately 2mm. The second imaging system is a catadioptric camera composed of an omnidirectional mirror and the same camera as before, equipped with a 12mm lens. The mirror is from 0-360.com, has a "proprietary shape" (supposedly close to hyperbolic) and a vertical field of view of 115°. As the camera is placed by hand and is therefore neither aligned with the rotational axis of the mirror nor placed in a focal point, this setup is to be considered as non-central.
The calibration object which will be used for these cameras is a planar chessboard pattern

Figure 6.1.1.: The camera with the fisheye lens calibrated in this section.



(a) The complete catadioptric system.

(b) Close-up of the omnidirectional mirror.

Figure 6.1.2.: The catadioptric system, composed of a camera with a 12mm lens and an omnidirectional mirror on top.

where each patch has a side length of 32mm. It is placed at different positions by hand with the intention to cover each pixel of the camera image three times.
The only input data for the calibration procedure are the pixel positions of the chessboard corners together with an estimation of their uncertainty, and the corresponding metric positions in the local coordinate system of the calibration planes. To detect these corners, first the FAST corner detector of Rosten and Drummond [RD05] is used to locate possible candidates. Likely chessboard corners are identified by the method proposed by Wang et al. [WWXX07] and the final position is determined by the subpixel refinement procedure introduced by Lucchese and Mitra in [LM02]. They state that chessboard corners can be determined with an accuracy of approximately 0.1 pixels, which defines the standard deviation σ_p of the corner noise for these experiments. A procedure to automatically associate the resulting image positions with calibration plane corner positions is described in [Hil13]. In case of failure or missing points, the corners need to be assigned manually.
After having detected the corners all the steps described in the last chapter will be executed, starting from the initial calibration method of Section 5.1, continuing with the complete calibration procedure of Section 5.2 and finishing with the surface model construction of Section 5.3.
Some exemplary images used for the calibration are shown in Figure 6.1.3. Overall, 18 images are utilized for the fisheye camera and 84 for the catadioptric system. They are placed with the goal of covering each pixel three times. Additionally, the calibration board is positioned once with the explicit purpose to serve as the virtual image plane. It is then placed several times again in the same image region, further away from the camera and with different orientations. This ensures that the initial calibration step delivers good starting values for the complete calibration procedure. All additional views are selected by the amount of overlap with the currently calibrated area. The proposed procedure is able to calibrate all views automatically without any user intervention.
For these real systems, no ground truth data, e.g. calibration plane poses, is available. To evaluate the quality of the results, only the residual ray-point distance was used in the past ([RW12b, RW12a]). With the improved procedure of this work, also uncertainty information is available. This allows to give an estimate for the accuracy of the final model. The covariance matrix of a viewing ray $\mathbf{L}$ can be used to generate parameters of a ray $\mathbf{L}_{2\sigma}$ that lies 2 standard deviations away from the actual ray. Due to the singularity of the covariance matrix, this is done in reduced space. Subsequently, the corresponding positional and angular distances $d_{2\sigma}$ and $\varphi_{2\sigma}$ can be calculated similarly to the error values d_e and φ_e. The resulting mean values can be found in Table 6.1.1. Figures 6.1.4 and 6.1.5 show the corresponding profiles of the ray-point distance as well as the distributions of the $d_{2\sigma}$ and $\varphi_{2\sigma}$ for the fisheye camera and the catadioptric system, respectively. Remark: the values of $\overline{r}$ shown in the table are calculated only from those rays that were used in the bundle adjustment procedure. This explains the difference from the mean ray-point distances shown in the figures, which are calculated for all rays of the camera systems.
Generally, the results prove the high accuracy of the calibration procedure and the final surface model. Residual ray-point distances of $\overline{r} = 0.3733$mm and $\overline{r} = 0.1947$mm for the

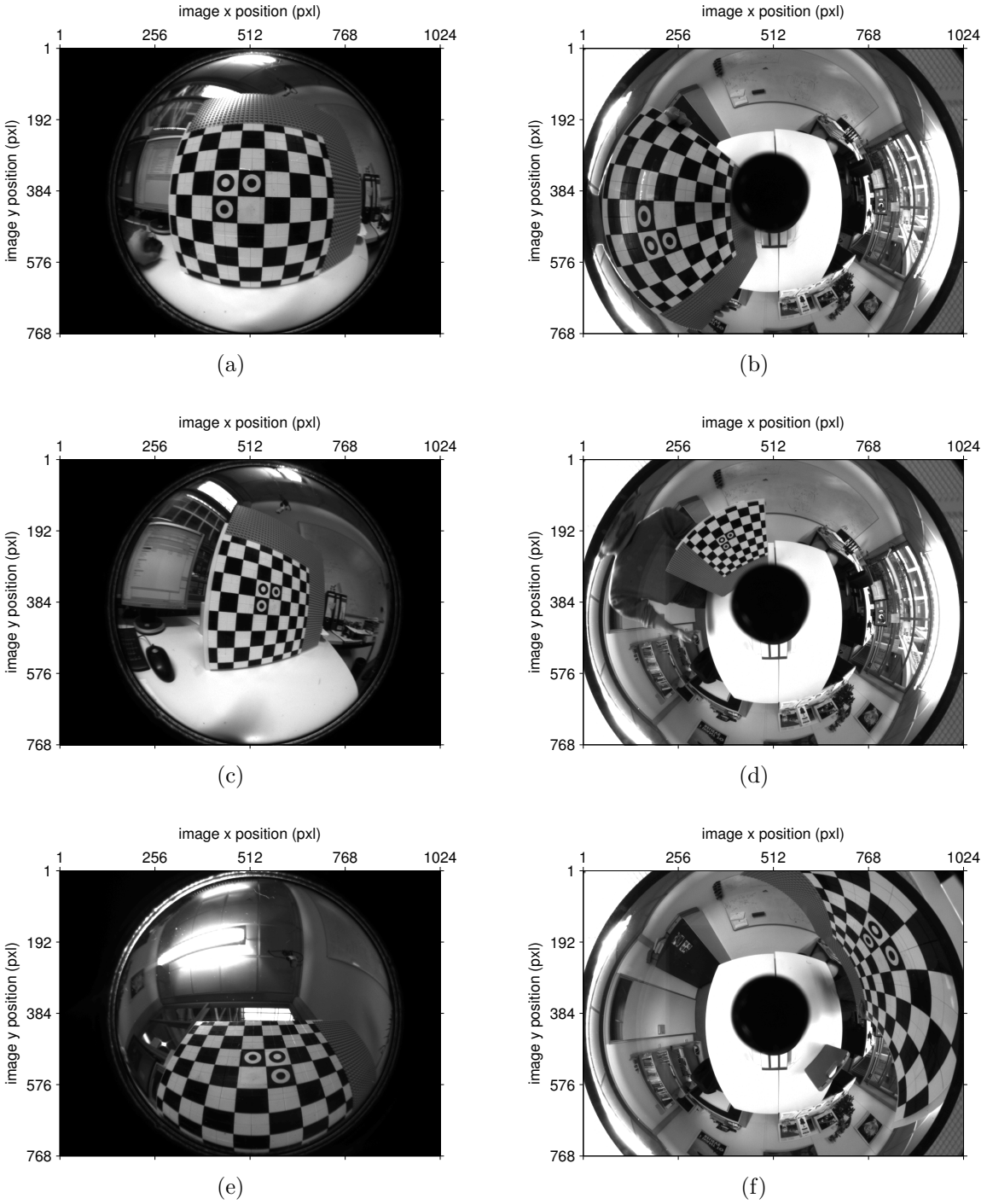
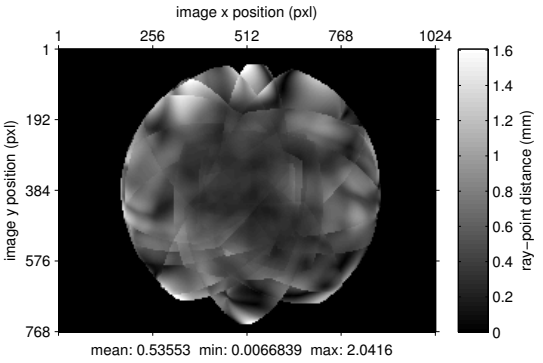
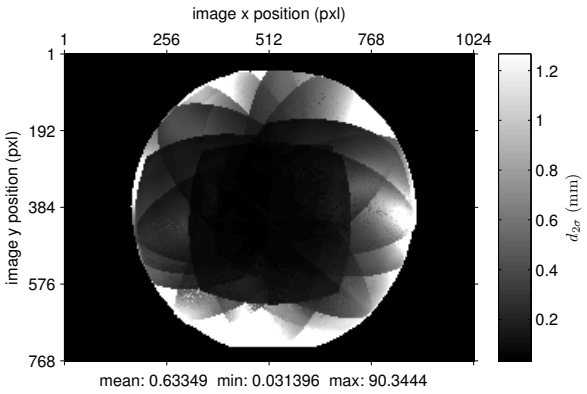


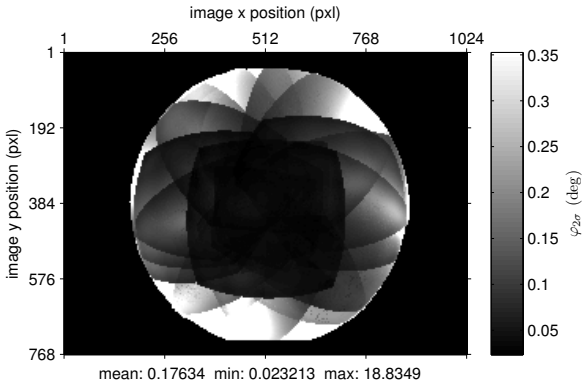
Figure 6.1.3.: Exemplary images used for calibrating the fisheye camera (left column) and the catadioptric system (right column). The images of the planes used as virtual image planes are shown in the first row.



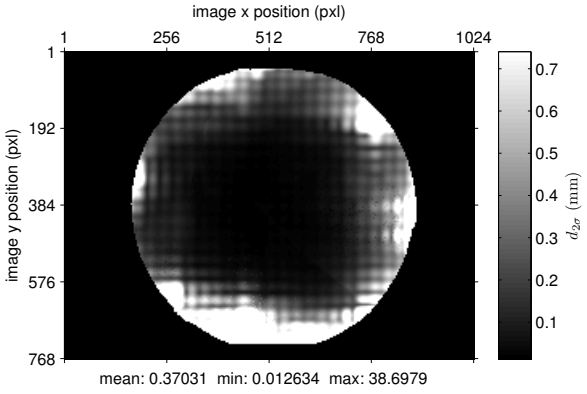
(a) Mean ray-point distances for every pixel.



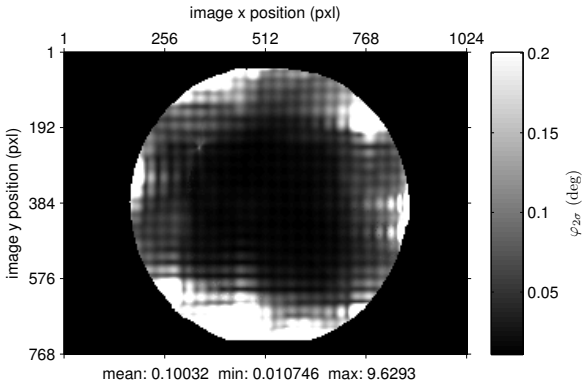
(b) $d_{2\sigma}$ of rays from calibration planes.



(c) $\varphi_{2\sigma}$ of rays from calibration planes.

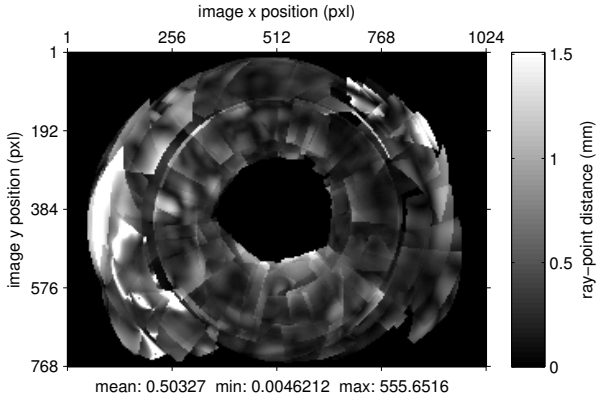


(d) $d_{2\sigma}$ of rays from surface model.

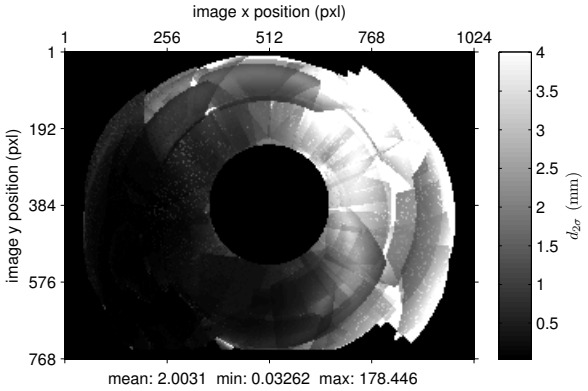


(e) $\varphi_{2\sigma}$ of rays from surface model.

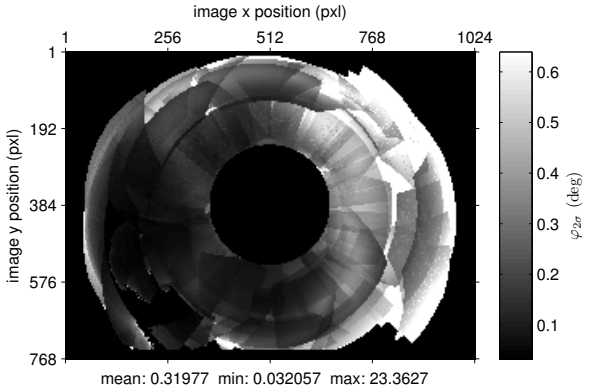
Figure 6.1.4.: Calibration of a real fisheye camera ($\sigma_p = 0.1$pxl). The first row depicts the distribution of the ray-point distances for every pixel. Below, the 2σ distances $d_{2\sigma}$ and $\varphi_{2\sigma}$ are shown. They are obtained by analyzing the uncertainties of the camera rays determined from the calibration planes (middle row) and from the surface model (last row). The data point position distance for constructing the surface model was 14 pixels, (25×23) control points were used for the spline approximation procedure.



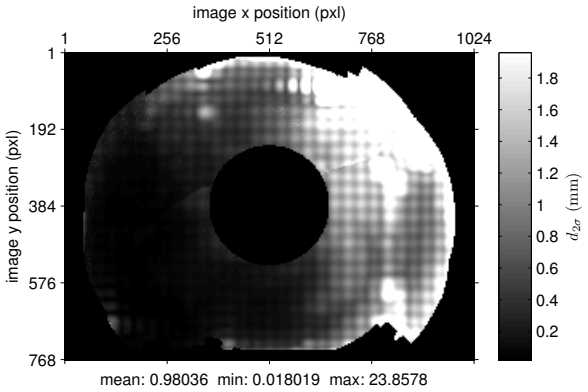
(a) Mean ray-point distances for every pixel.



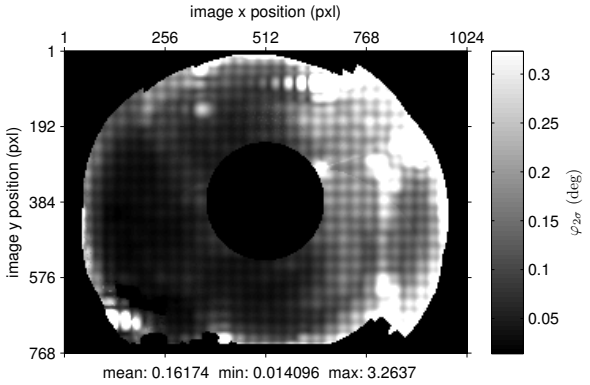
(b) $d_{2\sigma}$ of rays from calibration planes.



(c) $\varphi_{2\sigma}$ of rays from calibration planes.



(d) $d_{2\sigma}$ of rays from surface model



(e) $\varphi_{2\sigma}$ of rays from surface model.

Figure 6.1.5.: Calibration of a real catadioptric camera ($\sigma_p = 0.1$pxl). Ray-point distances are shown in the first row. Uncertainties of the viewing rays are determined from the calibration planes (middle row) and from the surface model (last row). It was constructed with a data point position distance of 14 pixels and by using (31×25) control points.

camera	$\overline{r}$	$\overline{d}_{2\sigma\_\text{planes}}$	$\overline{d}_{2\sigma\_\text{surf}}$	$\overline{\varphi}_{2\sigma\_\text{planes}}$	$\overline{\varphi}_{2\sigma\_\text{surf}}$
fisheye	0.3733mm	0.6335mm	0.3703mm	0.1763°	0.1003°
catadioptric	0.1947mm	2.0031mm	0.9804mm	0.3198°	0.1617°
panomorph	0.5509mm	6.8232mm	3.7509mm	0.4179°	0.2319°

Table 6.1.1.: Final results of the calibration process for the real camera systems (details on the panomorph camera can be found in Section 6.2).

fisheye camera and the catadioptric system, respectively, show that the poses of all calibration planes are determined with high accuracy. The mean values for the 2σ distances $\overline{d}_{2\sigma\_\text{planes}}$ determined directly from the calibration planes indicate that the positional errors can be expected to lie below 0.63mm for the fisheye and at 2mm for the catadioptric setup. The predicted angular accuracy $\overline{\varphi}_{2\sigma\_\text{planes}}$ is smaller than 0.2° for the fisheye and around 0.3° for the catadioptric camera.
The surface models are each created with a data point position distance of 14 pixels. (25×23) control points are used to represent the fisheye camera, (31×25) for the catadioptric system. The corresponding predicted errors of the surface models are smaller than those determined directly from the calibration planes for the reasons described in the last chapter. It is advisable to use the ratios of the mean Mahalanobis distances to correct the covariance matrices from the surface model and get a better uncertainty estimate.

6.2. Panomorph camera

In this section, a camera with an omnidirectional panomorph lens (see Section 3.2.1.1) will be calibrated. Its distortion characteristic is far from being radially symmetric, which is why standard fisheye models can not be used to describe it. The field of view is 182°, the camera images have a resolution of 2592×1944 pixels. The utilized planar calibration board has a printed chessboard pattern attached to it where each patch has a sidelength of 50mm. This board was placed by hand such as 3 hemispheres at a distance of approximately 75cm, 100cm and 125cm from the camera are covered. Overall, 218 calibration images are used. Some exemplary ones are shown in Figure 6.2.1, together with the coverage of the calibration boards. The surface model was created from data points with a distance of 28 pixels and by determining (35×25) spline control points. Again, a pixel noise of $\sigma_p = 0.1$pxl was assumed.
In [PGT12], Poulin-Gerard and Thibault determine the distribution of the inverse instantaneous field of view (IFOV^{-1}), which is a measure for the local angular resolution of a camera, for the used image region. A qualitative comparison of their results with the final calibration obtained in this work can be found in Figure 6.2.2. The profile lines along the horizontal symmetry axis of the lens show the same characteristics, having a peak on both

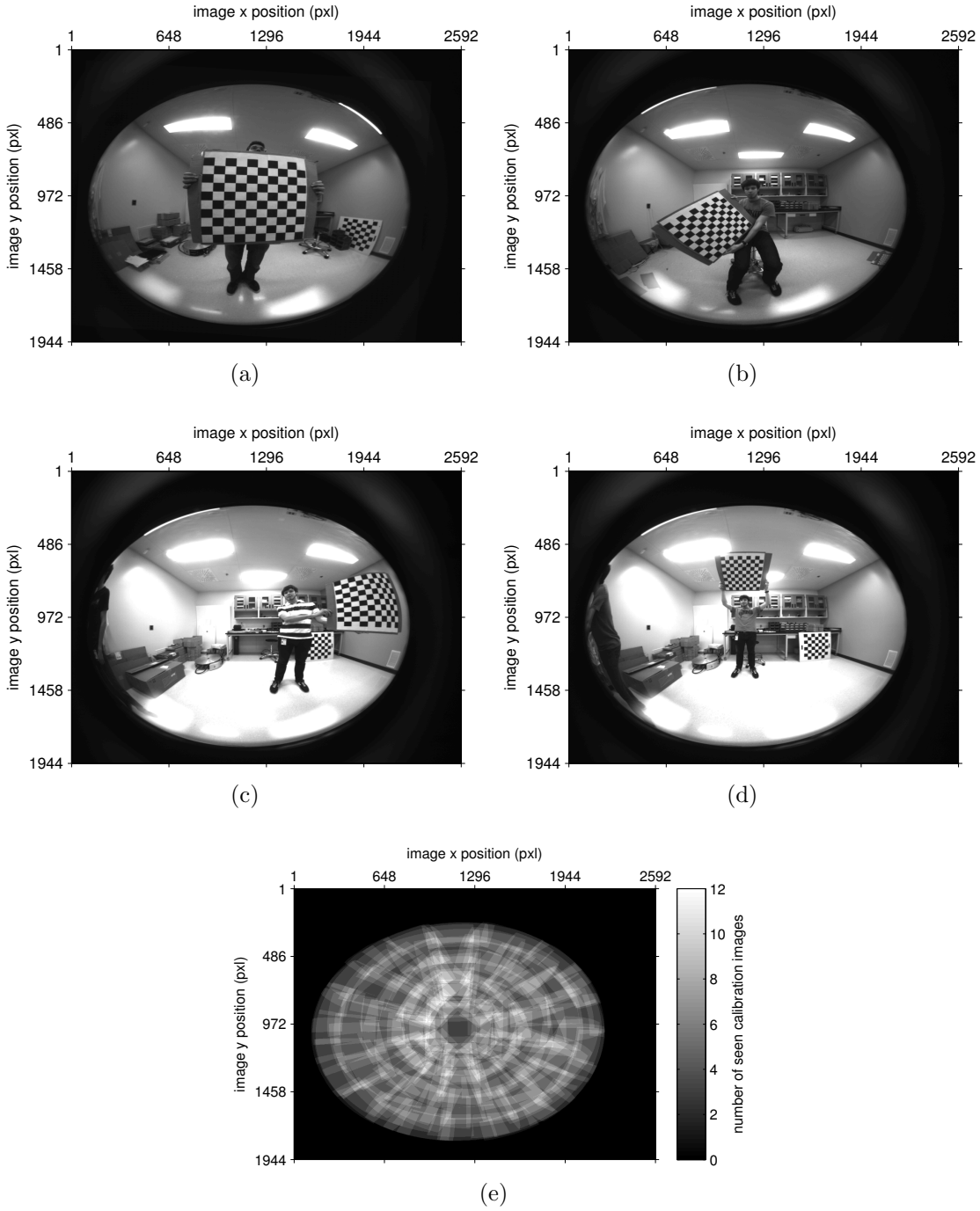


Figure 6.2.1.: Exemplary calibration image of the real panomorph camera camera (a-d). In (e), the image coverage of all utilized calibration boards is shown.

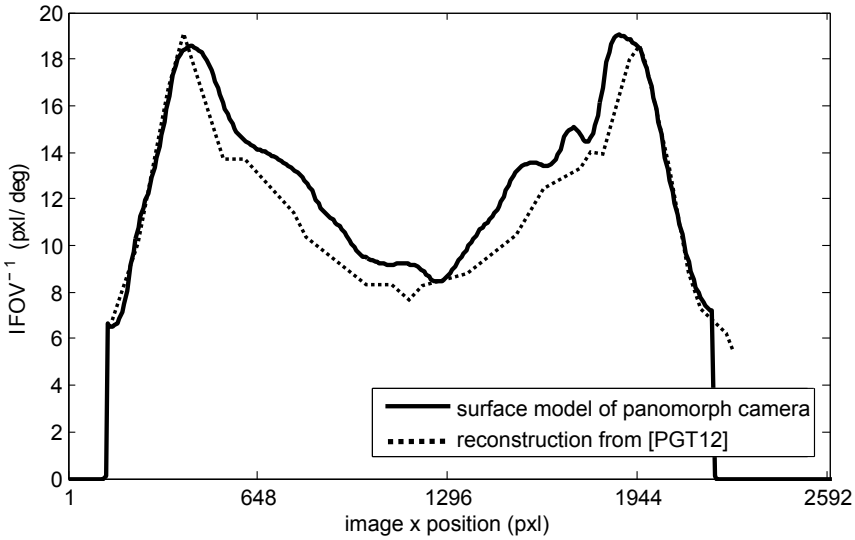
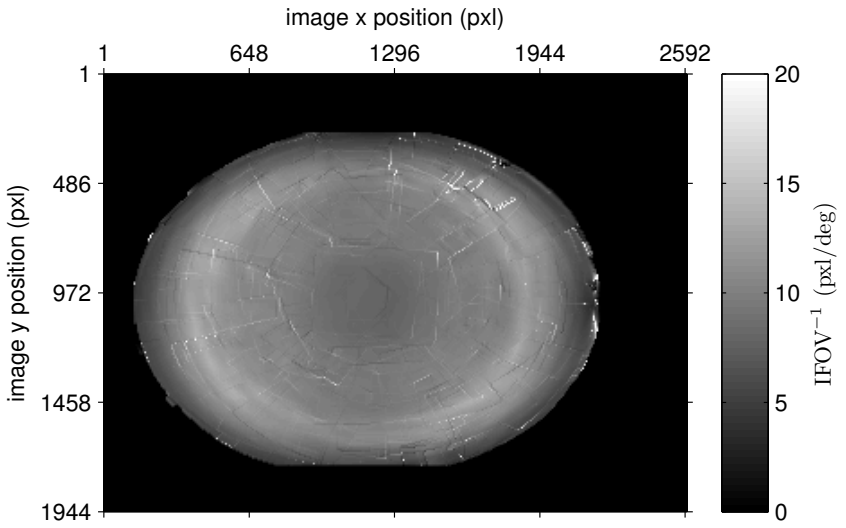


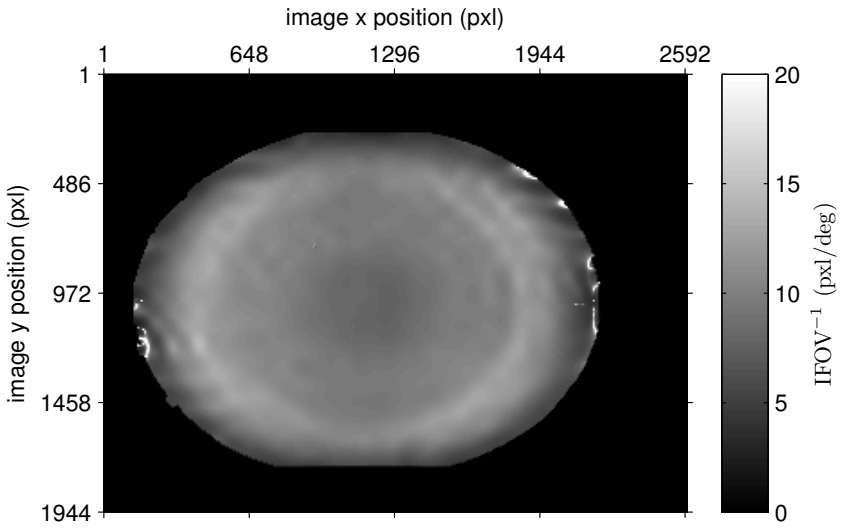
Figure 6.2.2.: Comparison of the IFOV^{-1} profiles along the horizontal symmetry axis of the panomorph camera.

sides of the optical center and values of similar sizes. The corresponding 2D profiles are shown in Figure 6.2.3. A qualitative comparison to the IFOV^{-1} profile in [PGT12] reveals that shape and values are very similar. This proves that the surface model parameterized by the procedure proposed in this work can be considered as a valid representation of this camera system.
The results of the calibration process can be found in Table 6.1.1. Due to the greater distance of the calibration planes to the camera it is not surprising that the residual ray-point distances are with 0.5509mm bigger than for the imaging systems calibrated in the previous section. This leads to increased uncertainties, which are expressed by the mean uncertainty distances $\overline{d}_{2\sigma\_\text{planes}}$ and $\overline{d}_{2\sigma\_\text{surf}}$. Here, especially the values for the ray positions are much higher than for the ordinary fisheye camera, having values of 6.82mm when determined directly from the planes and 3.75mm when using the spline surface. The mean angular distances $\overline{\varphi}_{2\sigma\_\text{planes}}$ and $\overline{\varphi}_{2\sigma\_\text{surf}}$ are much closer to the results for the fisheye camera, having values of 0.42° when obtained from the planes and 0.23° for the surface model. The corresponding profiles are shown in Figure 6.2.4. As before, it is advisable to adjust the covariance matrices of the surface model to get uncertainty estimates which are closer to the results obtained directly from the calibration planes.

This concludes the chapter on calibration of real cameras with the procedure proposed in the previous chapter. It was shown that the concepts are applicable in practice and serve to get consistent results. The resulting surface models accurately describe the imaging systems and deliver valuable uncertainty information.

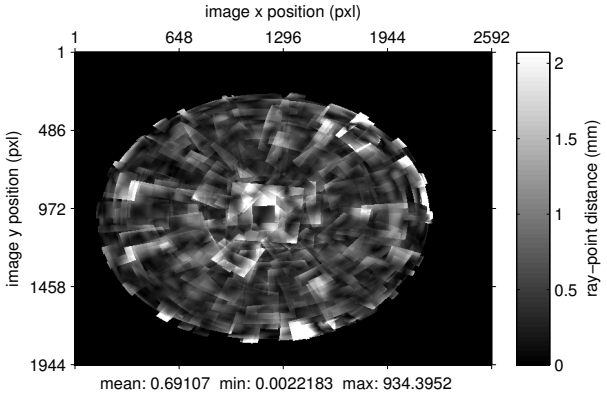


(a) IFOV^{-1} determined directly from the calibration planes.

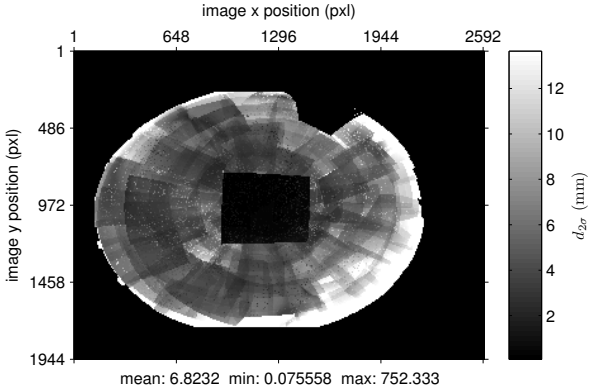


(b) IFOV^{-1} determined from the surface model.

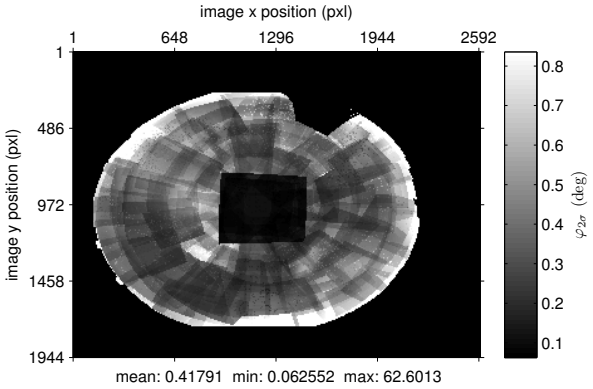
Figure 6.2.3.: IFOV^{-1} profiles of the real panomorph camera.



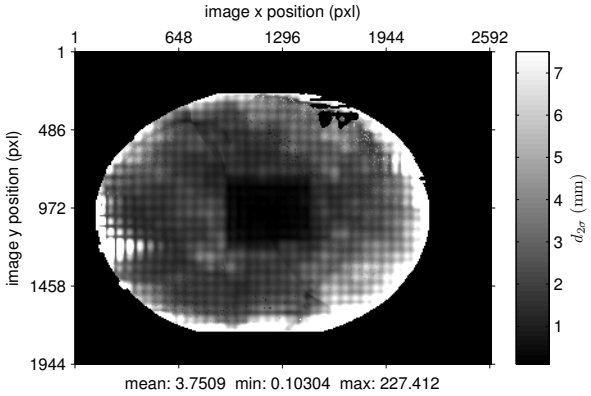
(a) Mean ray-point distances for every pixel.



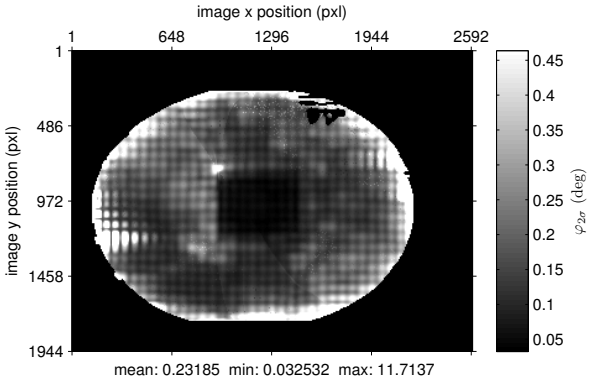
(b) $d_{2\sigma}$ of rays from calibration planes.



(c) $\varphi_{2\sigma}$ of rays from calibration planes.



(d) $d_{2\sigma}$ of rays from surface model



(e) $\varphi_{2\sigma}$ of rays from surface model.

Figure 6.2.4.: Calibration of real panomorph camera ($\sigma_p = 0.1$pxl). Ray-point distances are shown in the first row. Uncertainties of the viewing rays are determined from the calibration planes (middle row) and from the surface model (last row). It was constructed with a data point position distance of 28 pixels and by using (35×25) control points.

Chapter 7

Conclusion

Camera modeling and calibration are essential steps to be fulfilled before using digital imaging devices for measurement purposes. These steps are challenging and time-consuming since they require substantial knowledge about the internal structure of the utilized system, the available models that could adequately describe the system, and the corresponding calibration procedures. The main intention of this work was to introduce a model that can be used to describe different kinds of cameras, including central and non-central omnidirectional systems. Additionally, this work aimed to provide uncertainty information for the proposed model that can be used to assess the quality of a measurement result. Furthermore, this work set out to develop a calibration procedure that can easily be executed without requiring complex hardware equipment.

These goals describe the three main contributions of this work. In Chapter 4, the surface model was proposed as an uncertain continuous represention of the generic camera model. A spline surface in 6D Plücker space is utilized to provide the functions of general forward and back projection. It was verified by simulations that the model can describe both central and non-central camera systems with high accuracy. By using uncertainty propagation during spline creation, an uncertain surface can be generated that delivers covariance matrices together with the determined viewing ray parameters or pixel positions.

In Chapter 5, it was shown how the surface model can be calibrated by using sparse planar calibration patterns. The first step generates initial values with the assumption that the imaging setup is central. These values are refined in the subsequent optimization procedure, which makes explicit use of the covariance matrices for weighting purposes. Based on this intermediate result, which only calibrates a small part of the camera image, the calibrated area is expanded by iteratively adding further calibration planes. When sufficient information is available for the complete field of view, i.e. at least two 3D measurements per pixel are given, the surface model is created. Measurement uncertainties are taken into consideration when determining the uncertain rays, which serve as data

points for the 6D spline surface. This leads to the final surface model as an uncertain continuous representation of the camera system. Each step is analyzed by using simulated data, proving the applicability and evaluating the accuracy of the proposed concepts.
Chapter 6 applies the developed methods to the calibration of three different real cameras: a camera with a fisheye lens, a catadioptric system with an unknown omnidirectional mirror and a perspective camera with minor lens distortions, and a panomorph camera with a wide field of view and a distortion function that is rotationally asymmetric. The small residuals of the ray-point distances prove that a highly accurate result can be achieved with this method for very different types of cameras. Furthermore, consistent estimates of the uncertainties indicate the quality that can be expected when using this method in measurement applications.

7.1. Outlook

This work introduces the surface model and its calibration to simplify the process of using cameras for measurement tasks. Subsequent steps would include the application of the model to different tasks in computer vision. Exemplary applications are image rectification, visual odometry and structure from motion. The obtained uncertainties allow the results to be evaluated on a probabilistic level, which is advantageous when e.g. using them for data fusion.
During the development of the methods utilized in this work, computational complexity was not explicitly regarded. Some of the procedures are computationally expensive. For example, during the calibration of the panomorph camera, 218 different views of the calibration plane were utilized, leading to systems with more than 4000 equations. The matrix inversion needed to execute the least squares optimization during bundle adjustment requires a substantial amount of computation power. Another example for the computational demands of the proposed procedures is the utilization of a brute force approach to determine initial values for the forward projection procedure. Its application during e.g. classical structure from motion approaches, which necessitate projecting many 3D points to the camera image, would require a substantial amount of time. Consequently, reducing the computational load of the forward projection procedure is an essential next step to increase the usability of the surface model for real-time applications.
Currently, discontinuities of the camera mapping are not explicitly taken into account. For the simulated conic camera analyzed in this work, the problem was avoided by masking the corresponding area. If the discontinuities are linear and aligned with the parameter coordinate system of the spline surface, using multiple knots can solve this task. However, a generalization of this concept is not trivial. Therefore, further research is necessary to describe more complex setups, e.g. with multiple planar mirrors or multi-camera systems, with a single spline surface.

Appendix A

Appendix

A.1. Covariance matrix for inversion of uncertain spline surfaces

Section 5.1.1.1 elaborates how an uncertain spline surface can be inverted, i.e. how the parameters $(u_p, v_p)^T$ for the point on the spline that lies closest to an arbitrary point $\boldsymbol{P}$ can be determined. The functional relation at that point can be described as:

$$\begin{bmatrix} f_s \\ g_s \end{bmatrix} = \begin{bmatrix} (\boldsymbol{S}(u_p, v_p) - \boldsymbol{P})^T \\ (\boldsymbol{S}(u_p, v_p) - \boldsymbol{P})^T \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_u(u_p, v_p) \\ \boldsymbol{S}_v(u_p, v_p) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{A.1.1}$$

The corresponding covariance matrix is determined via implicit uncertainty propagation:

$$\Sigma_{u,v} = \mathrm{B}^{-1}\mathrm{A}\Sigma_{\boldsymbol{P}_p\boldsymbol{P}_p}\mathrm{A}^T\mathrm{B}^{-T}. \tag{A.1.2}$$

Matrix A contains the partial derivatives for all elements of these d-dimensional control points:

$$\mathrm{A} = \begin{bmatrix} \frac{\partial f_s}{\partial P_{s_u-p,s_v-p,0}} & \frac{\partial f_s}{\partial P_{s_u-p+1,s_v-p,0}} & \cdots & \frac{\partial f_s}{\partial P_{s_u,s_v,0}} & \frac{\partial f_s}{\partial P_{s_u-p,s_v-p,1}} & \cdots & \frac{\partial f_s}{\partial P_{s_u-p,s_v-p,d}} \\ \frac{\partial g_s}{\partial P_{s_u-p,s_v-p,0}} & \cdots & \cdots & \cdots & \cdots & \cdots & \frac{\partial g_s}{\partial P_{s_u-p,s_v-p,d}} \end{bmatrix}. \tag{A.1.3}$$

In matrix B, the partial derivatives of the parameters can be found:

$$\mathrm{B} = \begin{bmatrix} \frac{\partial f_s}{\partial u_p} & \frac{\partial f_s}{\partial v_p} \\ \frac{\partial g_s}{\partial u_p} & \frac{\partial g_s}{\partial v_p} \end{bmatrix}. \tag{A.1.4}$$

The derivatives contained in A are determined in the following way:

$$\frac{\partial f_s}{\partial P_{s_u-p,s_v-p,0}} = \frac{\partial \boldsymbol{S}^T(u_p,v_p)}{\partial P_{s_u-p,s_v-p,0}} \cdot \boldsymbol{S}_u(u_p,v_p) + (\boldsymbol{S}(u_p,v_p)-\boldsymbol{P})^T \cdot \frac{\partial \boldsymbol{S}_u(u_p,v_p)}{\partial P_{s_u-p,s_v-p,0}} \tag{A.1.5}$$

$$= \boldsymbol{S}_u^T(u_p,v_p) \cdot \frac{\partial \boldsymbol{S}(u_p,v_p)}{\partial P_{s_u-p,s_v-p,0}} + (\boldsymbol{S}(u_p,v_p)-\boldsymbol{P})^T \cdot \frac{\partial \boldsymbol{S}_u(u_p,v_p)}{\partial P_{s_u-p,s_v-p,0}}. \tag{A.1.6}$$

With

$$\boldsymbol{S}(u_p,v_p) = \sum_{i=s_u-p}^{s_u} \sum_{j=s_v-p}^{s_v} N_{i,p}(u_p) N_{j,p}(v_p) \boldsymbol{P}_{ij} \tag{A.1.7}$$

$$= N_{s_u-p,p}(u_p) N_{s_v-p,p}(v_p) \boldsymbol{P}_{s_u-p,s_v-p} + ... + N_{s_u,p}(u_p) N_{s_v,p}(v_p) \boldsymbol{P}_{s_u,s_v} \tag{A.1.8}$$

and therefore

$$\boldsymbol{S}_u(u_p,v_p) = N^{(1)}_{s_u-p,p}(u_p) N_{s_v-p,p}(v_p) \boldsymbol{P}_{s_u-p,s_v-p} + ... + N^{(1)}_{s_u,p}(u_p) N_{s_v,p}(v_p) \boldsymbol{P}_{s_u,s_v} \tag{A.1.9}$$

follows

$$\frac{\partial f_s}{\partial P_{s_u-p,s_v-p,0}} = N_{s_u-p,p}(u_p) N_{s_v-p,p}(v_p) \boldsymbol{S}_u^T \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + N^{(1)}_{s_u,p}(u_p) N_{s_v,p}(v_p) (\boldsymbol{S}-\boldsymbol{P})^T \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$= N_{s_u-p,p}(u_p) N_{s_v-p,p}(v_p) S_{u,0} + N^{(1)}_{s_u,p}(u_p) N_{s_v,p}(v_p)(S_0 - P_0). \tag{A.1.10}$$

The elements of B are calculated as

$$\frac{\partial f_s}{\partial u_p} = \frac{\partial}{\partial u_p} \left[(\boldsymbol{S}(u_p,v_p)-\boldsymbol{P})^T \cdot \boldsymbol{S}_u(u_p,v_p) \right] \tag{A.1.11}$$

$$= \boldsymbol{S}_u^T \boldsymbol{S}_u + (\boldsymbol{S}-\boldsymbol{P})^T \boldsymbol{S}_{uu} \tag{A.1.12}$$

$$\frac{\partial f_s}{\partial v_p} = \frac{\partial g_s}{\partial u_p} = \boldsymbol{S}_v^T \boldsymbol{S}_u + (\boldsymbol{S}-\boldsymbol{P})^T \boldsymbol{S}_{uv} \tag{A.1.13}$$

$$\frac{\partial g_s}{\partial v_p} = \boldsymbol{S}_v^T \boldsymbol{S}_v + (\boldsymbol{S}-\boldsymbol{P})^T \boldsymbol{S}_{vv}. \tag{A.1.14}$$

A.2. Data point conditioning for uncertain homography estimation

Section 5.1.2 shows how the homography H can be determined which describes the transformation of plane points and fulfills $\mathbf{p}_{1,i} = \mathrm{H}_v \mathbf{p}_{v,i}$. Data point conditioning is recommended for numerical reasons. It can be achieved by transforming the data points $\boldsymbol{p}_{1,i}$

and $\boldsymbol{p}_{v,i}$ such as they have zero mean and distances ≤ 1 from the origin (see [FW14], Section 5.9). With mean positions $\overline{\boldsymbol{p}}_{1,i} = (\overline{p}_{1,i_x}, \overline{p}_{1,i_y})^T$, $\overline{\boldsymbol{p}}_{v,i} = (\overline{p}_{v,i_x}, \overline{p}_{v,i_y})^T$ and d_{max_i} and d_{max_v} as the maximum distance of the corresponding points from the mean positions, the transformation matrices

$$\mathrm{T}_1 = \begin{bmatrix} 1 & 0 & -\overline{p}_{1,i_x} \\ 0 & 1 & -\overline{p}_{1,i_y} \\ 0 & 0 & d_{max_1} \end{bmatrix} \quad \text{and} \quad \mathrm{T}_v = \begin{bmatrix} 1 & 0 & -\overline{p}_{v,i_x} \\ 0 & 1 & -\overline{p}_{v,i_y} \\ 0 & 0 & d_{max_v} \end{bmatrix} \tag{A.2.1}$$

can be constructed. This leads to the conditioned points

$$\mathbf{p}'_{1,i} = \mathrm{T}_1 \mathbf{p}_{1,i} \quad \text{and} \quad \mathbf{p}'_{v,i} = \mathrm{T}_v \mathbf{p}_{v,i}. \tag{A.2.2}$$

By utilizing (5.1.14) and (5.1.20), the homography H'_v and the corresponding covariance matrix $\Sigma'_{\boldsymbol{h}_v \boldsymbol{h}_v}$ are determined. They need to be unconditioned to get the final results:

$$\mathrm{H}_v = \mathrm{T}_1^{-1} \mathrm{H}'_v \mathrm{T}_v \tag{A.2.3}$$

and with $\mathrm{T}_{vu} = \frac{\partial \mathrm{H}_v}{\partial \boldsymbol{h}'_v} = \mathrm{T}_v^T \otimes \mathrm{T}_1^{-1}$:

$$\Sigma_{\boldsymbol{h}_v \boldsymbol{h}_v} = \mathrm{T}_{vu} \Sigma'_{\boldsymbol{h}_v \boldsymbol{h}_v} \mathrm{T}_{vu}^T. \tag{A.2.4}$$

A.3. Jacobians for minimizing the ray-point distance

In this section, the Jacobians of the model function $\boldsymbol{f}_{v,i}$ for the ray-point distance are derived. The substitutions $r_t = \boldsymbol{r}_{v3}^T \boldsymbol{t}_v$ and $r_d = \boldsymbol{r}_{3_v}^T \boldsymbol{d}_i$ will be used whenever appropriate.

$$\boldsymbol{f}_{v,i} = \mathrm{R}_v^T (\boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i \boldsymbol{r}_{3_v}^T \boldsymbol{t}_v) \frac{1}{\boldsymbol{r}_{3_v}^T \boldsymbol{d}_i} - \mathrm{R}_v^T \boldsymbol{t}_v \tag{A.3.1}$$

$$\Leftrightarrow \quad \boldsymbol{f}_{v,i} = \mathrm{R}_v^T (\boldsymbol{r}_{3_v} \times \boldsymbol{m}_i + \boldsymbol{d}_i r_t) \frac{1}{r_d} - \mathrm{R}_v^T \boldsymbol{t}_v. \tag{A.3.2}$$

Partial derivatives of line parameters:

$$\begin{aligned}
\frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{d}_i} &= \frac{\partial}{\partial \boldsymbol{d}_i}\left(\mathrm{R}_v^T(\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i + \boldsymbol{d}_i r_t)\frac{1}{\boldsymbol{r}_{3_v}^T\boldsymbol{d}_i}\right) \\
&= \mathrm{R}_v^T\left(\mathrm{I}_3\frac{r_t}{r_d} - (\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i + \boldsymbol{d}_i r_t)\frac{\boldsymbol{r}_{3_v}^T}{r_d^2}\right) \\
&= \mathrm{R}_v^T\left(\mathrm{I}_3 r_d r_t - (\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i + \boldsymbol{d}_i r_t)\boldsymbol{r}_{3_v}^T\right)\frac{1}{r_d^2}
\end{aligned} \tag{A.3.3}$$

$$\begin{aligned}
\frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{m}_i} &= \frac{\partial}{\partial \boldsymbol{m}_i}\left(\mathrm{R}_v^T(\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i)\frac{1}{r_d}\right) \\
&= \frac{\partial}{\partial \boldsymbol{m}_i}\left(\mathrm{R}_v^T\mathsf{S}(\boldsymbol{r}_{3_v})\boldsymbol{m}_i\frac{1}{r_d}\right) \\
&= \mathrm{R}_v^T\mathsf{S}(\boldsymbol{r}_{3_v})\frac{1}{r_d}
\end{aligned} \tag{A.3.4}$$

$$\begin{aligned}
\frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{t}_v} &= \frac{\partial}{\partial \boldsymbol{t}_v}\left(\mathrm{R}_v^T(\boldsymbol{d}_i\boldsymbol{r}_{3_v}^T\boldsymbol{t}_v\frac{1}{r_d} - \boldsymbol{t}_v)\right) \\
&= \mathrm{R}_v^T\left(\frac{\boldsymbol{d}_i\boldsymbol{r}_{3_v}^T}{r_d} - \mathrm{I}_3\right).
\end{aligned} \tag{A.3.5}$$

The differentiation with respect to $\boldsymbol{R}_v$ is a little more elaborate, because $\boldsymbol{g}_i^v$ explicitly contains the third column of R_v, namely $\boldsymbol{r}_{3_v}$. Please see below for derivations of subterms.

$$\begin{aligned}
\frac{\partial \boldsymbol{f}_{v,i}}{\partial \boldsymbol{R}_v} &= \underbrace{\frac{\partial}{\partial \boldsymbol{R}_v}\left(\mathrm{R}_v^T(\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i)\right)}_{(A.3.7)}\frac{1}{r_d} + \underbrace{\frac{\partial}{\partial \boldsymbol{R}_v}\left(\mathrm{R}_v^T\boldsymbol{d}_i\boldsymbol{r}_{3_v}^T\boldsymbol{t}_v\right)}_{(A.3.8)}\frac{1}{r_d} \\
&\quad + \mathrm{R}_v^T(\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i + \boldsymbol{d}_i r_t)\underbrace{\frac{\partial}{\partial \boldsymbol{R}_v}\left(\frac{1}{\boldsymbol{r}_{3_v}^T\boldsymbol{d}_i}\right)}_{(A.3.9)} - \underbrace{\frac{\partial}{\partial \boldsymbol{R}_v}(\mathrm{R}_v^T\boldsymbol{t}_v)}_{(A.3.10)} \\
&= \mathrm{D}\mathrm{R}_v^T\mathsf{S}(\boldsymbol{m}_i)\frac{1}{r_d} + \mathrm{R}_v^T(\mathsf{S}(\boldsymbol{d}_i)r_t + \boldsymbol{d}_i\boldsymbol{r}_{3_v}^T\mathsf{S}(\boldsymbol{t}_v))\frac{1}{r_d} \\
&\quad - \mathrm{R}_v^T(\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i + \boldsymbol{d}_i r_t)\frac{\boldsymbol{r}_{3_v}^T\mathsf{S}(\boldsymbol{d}_i)}{r_d^2} - \mathrm{R}_v^T\mathsf{S}(\boldsymbol{t}_v) \\
&= \mathrm{R}_v^T\left(\mathsf{S}(\boldsymbol{d}_i)r_t + \boldsymbol{d}_i\boldsymbol{r}_{3_v}^T\mathsf{S}(\boldsymbol{t}_v) - (\boldsymbol{r}_{3_v}\times\boldsymbol{m}_i + \boldsymbol{d}_i r_t)\frac{\boldsymbol{r}_{3_v}^T\mathsf{S}(\boldsymbol{d}_i)}{r_d}\right)\frac{1}{r_d} \\
&\quad - \mathrm{R}_v^T\mathsf{S}(\boldsymbol{t}_v) + \mathrm{D}\mathrm{R}_v^T\mathsf{S}(\boldsymbol{m}_i)\frac{1}{r_d}
\end{aligned} \tag{A.3.6}$$

In the following derivations, subscripts will be omitted to improve readability.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{R}}\left(\mathrm{R}^T(\boldsymbol{r}_3 \times \boldsymbol{m})\right) &\overset{(A.3.11)}{=} \frac{\partial}{\partial \boldsymbol{R}}\left(\mathrm{D}\mathrm{R}^T\boldsymbol{m}\right)\\ &= \mathrm{D}\frac{\partial}{\partial \boldsymbol{R}}\mathrm{R}^T\boldsymbol{m}\\ &\overset{(2.1.27)}{=} \mathrm{D}\mathrm{R}^T\mathsf{S}(\boldsymbol{m})\end{aligned} \tag{A.3.7}$$

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{R}}(\mathrm{R}^T\boldsymbol{d}\,\boldsymbol{r}_3^T\boldsymbol{t}) &= \underbrace{\frac{\partial}{\partial \boldsymbol{R}}(\mathrm{R}^T\boldsymbol{d})}_{(A.3.10)}\, r_t + \mathrm{R}^T\boldsymbol{d}\,[0\ 0\ 1]\,\underbrace{\frac{\partial}{\partial \boldsymbol{R}}(\mathrm{R}^T\boldsymbol{t})}_{(A.3.10)}\\ &\overset{(2.1.27)}{=} \mathrm{R}^T\mathsf{S}(\boldsymbol{d})r_t + \mathrm{R}^T\boldsymbol{d}\,[0\ 0\ 1]\mathrm{R}^T\mathsf{S}(\boldsymbol{t})\\ &= \mathrm{R}^T\left(\mathsf{S}(\boldsymbol{d})r_t + \boldsymbol{d}\,\boldsymbol{r}_3^T\mathsf{S}(\boldsymbol{t})\right)\end{aligned} \tag{A.3.8}$$

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{R}}\left(\frac{1}{\boldsymbol{r}_3^T\boldsymbol{d}}\right) &= \frac{\partial}{\partial \boldsymbol{R}}(\boldsymbol{r}_3^T\boldsymbol{d})^{-1}\\ &= -\frac{\frac{\partial}{\partial \boldsymbol{R}}(\boldsymbol{r}_3^T\boldsymbol{d})}{r_d^2}\\ &= -\frac{\frac{\partial}{\partial \boldsymbol{R}}([0\ 0\ 1]\mathrm{R}^T\boldsymbol{d})}{r_d^2}\\ &= -\frac{[0\ 0\ 1]\frac{\partial}{\partial \boldsymbol{R}}(\mathrm{R}^T\boldsymbol{d})}{r_d^2}\\ &\overset{(A.3.10)}{=} -\frac{[0\ 0\ 1]\mathrm{R}^T\mathsf{S}(\boldsymbol{d})}{r_d^2}\\ &= -\frac{\boldsymbol{r}_3^T\mathsf{S}(\boldsymbol{d})}{r_d^2}\end{aligned} \tag{A.3.9}$$

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{R}}(\mathrm{R}^T\boldsymbol{d}) &= -e^{-\mathsf{S}(\boldsymbol{R})}\frac{\partial}{\partial \boldsymbol{R}}\left(\mathsf{S}(\boldsymbol{R})\boldsymbol{d}\right) \quad \left(\text{with } \mathrm{R}^T = \mathrm{R}^{-1} = e^{-\mathsf{S}(\boldsymbol{R})}\right)\\ &= -\mathrm{R}^T(-\mathsf{S}(\boldsymbol{d}))\\ &= \mathrm{R}^T\mathsf{S}(\boldsymbol{d})\end{aligned} \tag{A.3.10}$$

$$
\begin{aligned}
\mathrm{R}^T(\boldsymbol{r}_3 \times \boldsymbol{m}) &= \mathrm{R}^T \mathsf{S}(\boldsymbol{r}_3)\boldsymbol{m} \\
&= \begin{pmatrix} \boldsymbol{r}_1^T \\ \boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{pmatrix} \mathsf{S}(\boldsymbol{r}_3)\boldsymbol{m} = \begin{pmatrix} \boldsymbol{r}_1^T \mathsf{S}(\boldsymbol{r}_3) \\ \boldsymbol{r}_2^T \mathsf{S}(\boldsymbol{r}_3) \\ \boldsymbol{r}_3^T \mathsf{S}(\boldsymbol{r}_3) \end{pmatrix} \boldsymbol{m} \\
&= \begin{pmatrix} (\boldsymbol{r}_1 \times \boldsymbol{r}_3)^T \\ (\boldsymbol{r}_2 \times \boldsymbol{r}_3)^T \\ (\boldsymbol{r}_3 \times \boldsymbol{r}_3)^T \end{pmatrix} \boldsymbol{m} = \begin{pmatrix} -\boldsymbol{r}_2^T \\ \boldsymbol{r}_1^T \\ \boldsymbol{0}_3^T \end{pmatrix} \boldsymbol{m} \\
&= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{r}_1^T \\ \boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{pmatrix} \boldsymbol{m} \\
&= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathrm{R}^T \boldsymbol{m} \\
&= \mathrm{D}\mathrm{R}^T \boldsymbol{m} \quad \left(\text{with} \quad \mathrm{D} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)
\end{aligned}
\tag{A.3.11}
$$

Bibliography

[ATR10] Amit Agrawal, Yuichi Taguchi, and Srikumar Ramalingam. Analytical Forward Projection for Axial Non-Central Dioptric & Catadioptric Cameras. In *Proceedings of the European Conference On Computer Vision (ECCV)*, pages 129–143, 2010.

[ATR11] Amit Agrawal, Yuichi Taguchi, and Srikumar Ramalingam. Beyond Alhazen's problem: Analytical projection model for non-central catadioptric cameras with quadric mirrors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2993–3000. IEEE, June 2011.

[BN99] Simon Baker and Shree Nayar. A Theory of Single-Viewpoint Catadioptric Image Formation. *International Journal of Computer Vision*, 35(2):175–196, 1999.

[Bro66] Duane Brown. Decentering Distortion of Lenses. *Photogrammetric Engineering*, 32(3):444–462, 1966.

[BW70] Max Born and Emil Wolf. *Principles of Optics*. Pergamon Press, 4th editio edition, 1970.

[CBK80] Nai-yung Chen, John R. Birk, and Robert B. Kelley. Estimating Workpiece Pose Using the Feature Points Method. *IEEE Transactions on Automatic Control*, 25(6):1027–1041, 1980.

[CLSC92] G. Champleboux, S. Lavallée, P. Sautot, and P. Cinquin. Accurate Calibration of Cameras and Range Imaging Sensors: the NPBS Method. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1552–1557, Nice, 1992.

[Con19] Alexander Conrady. Decentred Lens-Systems. *Monthly notices of the Royal Astronomical Society*, 79:384–390, 1919.

[DK00] Steven Derrien and Kurt Konolige. Approximating a Single Viewpoint in Panoramic Imaging Devices. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, number April, pages 3931–3938, 2000.

[DMW07] Aubrey K. Dunne, John Mallon, and Paul F. Whelan. A Comparison of New Generic Camera Calibration with the Standard Parametric Approach. In *IAPR Conference on Machine Vision Applications, MVA2007*, pages 3–6, Tokyo, 2007.

[DMW10] Aubrey K. Dunne, John Mallon, and Paul F. Whelan. Efficient generic calibration method for general cameras with single centre of projection. *Computer Vision and Image Understanding*, 114(2):220–233, February 2010.

[Foe10] Wolfgang Foerstner. Minimal Representations for Uncertainty and Estimation in Projective Spaces. In *Proceedings of the 10th Asian Conference on Computer Vision (ACCV)*, volume 6493 of *Lecture Notes in Computer Science*, pages 619–632, Queenstown, 2010.

[FW14] Wolfgang Förstner and Bernhard P. Wrobel. *Photogrammetric Computer Vision; Geometry, Orientation and Reconstruction.* 2014.

[GD00] Christopher Geyer and Kostas Daniilidis. A Unifying Theory for Central Panoramic Systems and Practical Implications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 445–461, 2000.

[GD01] Christopher Geyer and Kostas Daniilidis. Catadioptric Projective Geometry. *International Journal of Computer Vision*, 45(3):223–243, 2001.

[GN01] Michael D. Grossberg and S.K. Nayar. A general imaging model and a method for finding its parameters. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 108–115, 2001.

[GN05] Michael D. Grossberg and Shree K. Nayar. The Raxel Imaging Model and Ray-Based Calibration. *International Journal of Computer Vision*, 61(2):119–137, 2005.

[GTK88] Keith D. Gremban, Charles E. Thorpe, and Takeo Kanade. Geometric Camera Calibration Using Systems of Linear Equations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 562–567, Philadelphia, 1988.

[Han11] Tobias Hanning. *High Precision Camera Calibration.* Vieweg+Teubner, Wiesbaden, 2011.

[Hil13] Robert Hillmann. Automatische Schachbretterkennung in Bildern beliebiger Kameras zur Implementierung eines benutzerfreundlichen Kalibrierframeworks. Technical report, 2013.

[HS97] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1106–1112. IEEE Comput. Soc, 1997.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry.* Cambridge University Press, 2nd edition, 2003.

[JU97] Simon J. Julier and Jeffrey K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *Proceedings of AeroSense: The 11th Int. Symposium on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[KB04] Juho Kannala and Sami Brandt. A generic camera calibration method for fish-eye lenses. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 10–13, 2004.

[Lhu08] Maxime Lhuillier. Automatic scene structure and camera motion using a catadioptric system. *Computer Vision and Image Understanding*, 109(2):186–203, February 2008.

[LM02] L. Lucchese and S.K. Mitra. Using saddle points for subpixel feature detection in camera calibration targets. *Asia-Pacific Conference on Circuits and Systems*, pages 191–195, 2002.

[MA10] Pedro Miraldo and Helder Araujo. Improving the Resolution of the Generic Camera Model By Means of a Parametric Representation. In *Proceedings of the 9th Portuguese Conference on Automatic Control (CONTROLO)*, Coimbra, 2010.

[MA11] Pedro Miraldo and Helder Araujo. Point-based Calibration Using a Parametric Representation of the General Imaging Model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2304–2311, Barcelona, 2011.

[MA13] Pedro Miraldo and Helder Araujo. Calibration of smooth camera mod-

els. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2091–103, September 2013.

[MAQ11] Pedro Miraldo, Helder Araujo, and Joao Queiró. Unique Solution for the Estimation of the Plücker Coordinates Using Radial Basis Functions (Technical Report). Technical report, 2011.

[MBK81] H. A. Martins, John R. Birk, and Robert B. Kelley. Camera Models Based on Data from Two Calibration Planes. *Computer Graphics and Image Processing*, 17:173–180, 1981.

[MP04] Branislav Micusík and Tomás Pajdla. Autocalibration & 3D Reconstruction with Non-central Catadioptric Cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2004.

[NM65] John A. Nelder and Roger Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[PBSG11] Luis Puig, J. Bermúdez, Peter Sturm, and J.J. Guerrero. Calibration of Omnidirectional Cameras in Practice. A Comparison of Methods. *Computer Vision and Image Understanding*, 116(1):120–137, September 2011.

[PGPTD10] Anne-Sophie Poulin-Girard, Jocelyn Parent, Simon Thibault, and Pierre Désaulniers. Dedicated testing setup for panoramic lenses. *Proceedings of SPIE*, 7786, 2010.

[PGT12] Anne-Sophie Poulin-Girard and Simon Thibault. Optical testing of panoramic lenses. *Optical Engineering*, 51(5), 2012.

[Ple03] Robert Pless. Using Many Cameras as One. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 587–593, 2003.

[PT97] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer, 2nd edition, 1997.

[RD05] E Rosten and T Drummond. Fusing points and lines for high-performance tracking. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1508–1515, 2005.

[RLS04] Srikumar Ramalingam, Suresh K. Lodha, and Peter Sturm. A Generic Structure-from-Motion Algorithm for Cross-Camera Scenarios. In *Workshop*

on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS), volume 103, pages 175–186, 2004.

[RSL05] Srikumar Ramalingam, Peter Sturm, and Suresh K. Lodha. Towards Complete Generic Camera Calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[RW12a] Dennis Rosebrock and Friedrich M. Wahl. Complete Generic Camera Calibration and Modeling Using Spline Surfaces. In *Proceedings of the 11th Asian Conference on Computer Vision (ACCV)*, Daejeon, Korea, 2012.

[RW12b] Dennis Rosebrock and Friedrich M. Wahl. Generic camera calibration and modeling using spline surfaces. In *IEEE Intelligent Vehicles Symposium*, Alcalá de Henares, Spain, 2012.

[SGN01] Rahul Swaminathan, Michael D. Grossberg, and Shree K. Nayar. Caustics of Catadioptric Cameras. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2–9, 2001.

[SGN06] Rahul Swaminathan, Michael D. Grossberg, and Shree K. Nayar. Non-Single Viewpoint Catadioptric Cameras: Geometry and Analysis. *International Journal of Computer Vision*, 66(3):211–229, March 2006.

[SM99] Peter Sturm and S.J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 432–437, 1999.

[SMS06] D. Scaramuzza, A. Martinelli, and R. Siegwart. A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion. *Proceedings of the Fourth IEEE Conference on Computer Vision Systems (ICVS)*, page 45, 2006.

[SR04] Peter Sturm and Srikumar Ramalingam. A Generic Concept for Camera Calibration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 1–13, 2004.

[SRL05] Peter Sturm, Srikumar Ramalingam, and Suresh K. Lodha. On calibration, structure-from-motion and multi-view geometry for panoramic camera models. In *Panoramic Photogrammetry Workshop*, Berlin, 2005.

[Stu10] Peter Sturm. Camera Models and Fundamental Concepts Used in Geomet-

ric Computer Vision. *Foundations and Trends in Computer Graphics and Vision*, 6(1-2):1–183, 2010.

[Tsa87] Roger Tsai. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, 1987.

[WM91] Guo-Qing Wei and Song De Ma. Two Plane Camera Calibration: A Unified Model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 133–138, 1991.

[WM93] Guo-Qing Wei and Song De Ma. A Complete Two-plane Camera Calibration Method and Experimental Comparisons. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 439–446, Berlin, 1993.

[WWXX07] Zhongshi Wang, Wei Wu, Xinhe Xu, and Dingyu Xue. Recognition and location of the internal corners of planar checkerboard calibration pattern image. *Applied Mathematics and Computation*, 185(2):894–906, 2007.

[Zha02] Zhengyou Zhang. A Flexible New Technique for Camera Calibration (Technical Report). Technical Report 11, 2002.